3
L. 2800

CORSO PRATICO DI PROGRAMMAZIONE PER LAVORARE E DIVERTIRSI COL COMPUTER

ISTITUTO GEOGRAFICO DE AGOSTINI

CORSO PRATICO DI PROGRAMMAZIONE
PER LAVORARE E DIVERTIRSI COL COMPUTER

Pubblicazione a fascicoli settimanali
edita dall'Istituto Geografico De Agostini

volume I - fascicolo 3

Direttori: Achille Boroli - Adolfo Boroli

Direzione editoriale: Mario Nilo; *settore fascicoli:* Jason Vella

Redazione dell'edizione italiana a cura della:
Logical Studio Communication
Traduzione dall'inglese a cura di: Daniel Quinn

Coordinamento grafico: Otello Geddo
Coordinamento fotografico a cura del Centro Iconografico dell'Istituto Geografico De Agostini
Direzione: Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5
Redazione: Milano (20149), via Mosè Bianchi 6 - tel. (02) 4694451

Programma di abbonamento. Condizioni di abbonamento all'intera opera in 52 fascicoli, completa di copertine e di risguardi per la confezione dei 6 volumi dell'opera:
a) in un unico versamento anticipato di L. 180 000 in Italia, L. 225 000 all'estero;
b) in 4 versamenti trimestrali consecutivi e anticipati di L. 45 250 ciascuno.
La forma di abbonamento *b* è ammessa soltanto in Italia.
Agli abbonati all'intera opera sono riservato in dono "2 cassette di videogiochi" oppure, in alternativa, "5 cassette da registrare" (Aut. Min. conc.).
I versamenti possono essere effettuati a mezzo assegno bancario oppure sul c/c postale n. 111286 intestato all'Istituto Geografico De Agostini - Novara.
Amministrazione, abbonamenti e servizio arretrati: Istituto Geografico De Agostini - Novara (28100), via Giovanni da Verrazano 15 - tel. (0321) 471201-5.
Copertine e risguardi per i volumi dell'opera saranno messi in vendita a L. 6000 la copia (L. 7500 all'estero).
Le copie arretrate saranno disponibili per un anno dal completamento dell'opera e potranno essere prenotate nelle edicole o direttamente presso l'Editore. Per i fascicoli arretrati, trascorse 12 settimane dalla loro pubblicazione, è applicato un sovrapprezzo di L. 400 sul prezzo di copertina in vigore al momento dell'evasione dell'ordine. Spedizione contro rimessa di pagamento anticipato; non vengono effettuate spedizioni contrassegno.
L'Editore si riserva la facoltà di modificare il prezzo nel corso della pubblicazione, se costretto da mutate condizioni di mercato.

Registrato presso il Tribunale di Novara n. 11 in data 19-5-1984.
Direttore responsabile: Emilio Bucciotti

Spedizione in abbonamento postale Gruppo II/70 (Autorizzazione della Direzione provinciale delle PP.TT. di Novara).
Distribuzione A. & G. Marco - Milano, via Fortezza 27 - tel. (02) 2526.
Pubblicazione a fascicoli settimanali. Esce il martedì.
Stampato in Italia - I.G.D.A. Officine Grafiche, Novara - 068411.

Referenze dei disegni e delle fotografie:
Copertina: Dave King. Pagg. 65-67 Malcolm Harrison. Pagg. 68-74, disegni Peter Bentley; fotografie John Darling. Pagg. 75-79 Dave King. Pagg. 80-83 Julek Heller. Pagg. 84-91, disegni Colin Hadley; fotografie John Darling. Pagg. 92-96 Dave King.

INPUT È STUDIATA APPOSITAMENTE PER:

Lo SPECTRUM della Sinclair (versioni 16K e 48K), il COMMODORE 64, l'ELECTRON ed il BBC della ACORN, il DRAGON 32.

Comunque, molti dei programmi e dei testi sono adatti anche per: lo ZX81 della SINCLAIR, il COMMODORE VIC 20 ed il TANDY COLOUR COMPUTER con 32K ed il BASIC esteso.

I seguenti simboli identificano i programmi o le spiegazioni adatte a ciascun computer:

 SPECTRUM

 COMMODORE 64

 ELECTRON e BBC

 DRAGON 32

 ZX81

 VIC 20

 TANDY TRS80 COLOUR COMPUTER

I LINGUAGGI DI BASSO LIVELLO

- CHE COS'È IL CODICE MACCHINA?
- VANTAGGI DEL CODICE MACCHINA RISPETTO AL **BASIC**
- CAPIRE GLI OPCODE E IL LINGUAGGIO ASSEMBLY

Azioni più rapide e movimenti più scorrevoli sullo schermo sono solo due dei vantaggi che si hanno programmando i giochi in codice macchina. Prima, però, occorre sapere come il computer interpreta il codice macchina

La maggior parte dei possessori di home computer trovano il BASIC più che adatto alle loro necessità: lo si può usare per semplici giochi o per complessi programmi applicativi. Quel che più conta, in ogni caso, è che il linguaggio BASIC è facile da imparare, il suo uso è molto elastico e lo si può adattare a macchine diverse.

Gli unici inconvenienti sono la lentezza d'esecuzione e la relativamente cospicua occupazione di memoria: i programmi BASIC divorano la memoria!

Quando si vogliono scrivere dei buoni giochi, ci si rende conto quanto i movimenti generati dal BASIC siano "a scatti". Inoltre, è possibile controllare una sola funzione per volta: se si spara un proiettile, ad esempio, tutte le altre operazioni devono essere momentaneamente sospese, anche se solo per brevi attimi. E se nel programma compaiono molti cicli FOR ... NEXT, l'esecuzione può diventare esasperatamente lenta.

È per queste ragioni che un programmatore con serie intenzioni lascia alle proprie spalle il BASIC, per dedicarsi al codice macchina, che consente un maggior controllo sul computer.

CHE COS'È IL CODICE MACCHINA?

Una delle ragioni della lentezza del BASIC deriva dal fatto che si tratta di un linguaggio *di alto livello*: ciò significa che vengono impiegate parole ed espressioni aritmetiche vicine al linguaggio umano.

Ma il computer non 'pensa' in inglese, né in alcun altro linguaggio umano. Per esser sinceri, un computer non *pensa*, né *capisce* niente, ma opera secondo impulsi elettrici che rappresentano numeri. Il linguaggio macchina, dunque, è unicamente composto dall'equivalente di puri e semplici numeri. Perciò, quando si programma in codice macchina, non ci si attende certo che il computer parli la nostra lingua: siamo noi, in un certo senso, a esprimerci nella sua. Tanto per fare un esempio (in questo caso per lo Spectrum), il codice macchina ha questo aspetto:

```
B9 28 08
```

mentre in BASIC ciò equivale a:

```
100 IF A=C THEN GOTO 190
```

Il codice macchina è, dunque, una serie di numeri a due cifre. Le lettere che talvolta appaiano (come la B nell'esempio) sono anch'esse da considerarsi numeri: infatti, nella numerazione *esadecimale* B rappresenta 11.

Questi numeri esadecimali vengono depositati direttamente nella memoria del computer e con essi si possono rappresentare codici operativi, dati, numeri, lettere o indirizzi di memoria.

Il computer è in grado di distinguere se

si tratta di un'istruzione o di un indirizzo o di un dato, soltanto in base all'ordine con cui compaiono i codici. Per esempio, il primo numero, in un programma, deve sempre corrispondere a quello di un'istruzione: se per errore inserissimo per primo un codice relativo a un dato, il computer tenterebbe di eseguirlo come se si trattasse di un'istruzione, causando un caos indescrivibile in tutto ciò che segue.

È quindi necessaria una meticolosa cura nell'immissione di tali codici: un solo errore e il programma non funziona o, peggio ancora, viene cancellato completamente dalla memoria.

VANTAGGI DEL CODICE MACCHINA

Durante l'immissione di un programma BASIC, il computer deve interpretare ciascuna linea, prima di poterla eseguire. Questo procedimento è lento e laborioso: generalmente, a un'istruzione BASIC equivale tutta una serie di istruzioni in codice macchina e, poiché le operazioni sono spesso abbastanza complesse e ripetitive, i tempi si allungano di conseguenza.

Quando si lancia un programma scritto interamente in BASIC, ciascuna linea viene interpretata e quindi eseguita, prima di poter passare a quella successiva. Nel caso dei cicli FOR ... NEXT, le cose si aggravano ulteriormente: a ogni passaggio, le linee devono essere nuovamente interpretate, con grande spreco di tempo e di energia.

Per sommi capi, il procedimento seguito dal BASIC si può riassumere così:

- Estrazione dell'istruzione
- Trasposizione dell'istruzione in una serie di codici operativi del linguaggio macchina
- Esecuzione di un codice operativo alla volta
- Passaggio alla successiva istruzione

Non è difficile rendersi conto di quanto sia lento un simile modo di procedere.

È possibile *compilare* i programmi BASIC (ossia convertire in una sola passata tutto il programma in codice macchina, prima di eseguire un RUN), mediante appositi programmi chiamati, appunto, *compilatori*.

Ma, sebbene vi sia un notevole guadagno, i programmi risultano ancora lunghi e più lenti del necessario.

Usando il codice macchina, invece, il processo di conversione, pur sempre necessario per generare gli impulsi elettrici comprensibili per il microprocessore, è ridotto al minimo: ogni numero corrisponde a una particolare configurazione di segnali. Niente complicati procedimenti interpretativi, quindi, ma una corrispondenza univoca tra istruzioni e codici.

In altre parole, i programmi in codice macchina sono molto più corti e veloci di quelli scritti in BASIC

Per quantificare la diversa velocità d'esecuzione dei due linguaggi, si usi il seguente programma, nel quale una stessa operazione viene eseguita prima in BASIC e poi in codice macchina:

```
1 MODE 6
10 FOR T=&1000 TO &101C
20 READ A:?T=A
30 NEXT
40 PRINT"LA LENTEZZA DEL BASIC..."
60 FOR T=&6100 TO &7FFF
70 ?T=?(T+&2000)
80 NEXT
90 CLS
100 PRINT"LA VELOCITÀ DEL CODICE
    MACCHINA"
105 DUMMY=INKEY(100)
110 CALL &1000
120 END
200 DATA 162,96,160,0,185,0,128,153,0,96,
    200,208,247,232,138,24,105,32,141,6,16,
    142,9,16,224,128,208,230,96
```

```
10 CLEAR 29999
20 FOR n=30000 TO 30011
30 READ a
40 POKE n,a
50 NEXT n
60 PRINT "Questo è il BASIC"
70 FOR n=16384 TO 22527
80 POKE n,PEEK (n-16384)
90 NEXT n
100 CLS
110 PRINT "Adesso... il CODICE MACCHINA"
120 PAUSE 100
130 RANDOMIZE USR 30000
140 STOP
150 DATA 33,0,0,17,0,64,1,0,24,237,176,201
```

```
10 CLEAR 200,31000
20 DEFUSR0=31000
30 FOR N=31000 TO 31015
40 READ A
50 POKE N,A
60 NEXT
70 CLS0
80 PRINT@0,"QUESTO È IL BASIC!"
90 FOR N=1 TO 500:NEXT
100 FOR N=1056 TO 1535
110 POKE N,PEEK(N+34000)
120 NEXT
130 FOR N=1 TO 1000:NEXT
140 CLS0
150 PRINT@0,"QUESTO È IL CODICE
    MACCHINA!"
160 FOR N=1 TO 1000: NEXT
170 N=USR0(0)
180 FOR N=1 TO 2000:NEXT
190 DATA 206,136,240,142,4,32,166,192,
    167,128,140,6,0,38,247,57
```

```
10 PRINT "  LA LENTEZZA DEL
    BASIC...":POKE 53280,7:POKE 53281,7
20 FOR Z=0 TO 959:POKE 55336+Z,1:
    POKE 1064+Z,46:NEXT Z
30 FOR Z=1 TO 1000:NEXT Z
40 PRINT " "
50 FOR Z=1 TO 1000:NEXT Z
100 PRINT" LA VELOCITÀ DEL CODICE
    MACCHINA":POKE 53280,10:POKE
    53281,10
105 FOR Z=1 TO 2000:NEXT Z
110 FOR Z=0 TO 19:READ X:POKE 832+Z,
    X:NEXT Z
120 SYS 832
130 DATA 162,0,169,46,157,40,4,157,0,5,157,
    0,6
135 DATA 157,232,6,232,208,241,96
```

```
10 PRINT"   LA LENTEZZA DEL
    BASIC...":POKE 36879,8
20 FOR Z=0 TO 483:POKE 7702+Z,46:
    NEXTZ
30 FOR Z=1 TO 1000:NEXT Z
40 PRINT " "
50 FOR Z=1 TO 1000:NEXT Z
100 PRINT" LA VELOCITÀ DEL CODICE
    MACCHINA"
```

```
105 FOR Z=1 TO 2000:NEXT Z
110 FOR Z=0 TO 13:READX:POKE 832+Z,
    X:NEXT Z
120 SYS 832
130 DATA 162,0,169,46,157,22,30,157,0,31,
    232,208,245,96
```

Il programma, in sé è poco interessante, consente tuttavia di apprezzare quanto la versione in codice macchina abbia una maggiore velocità di esecuzione.

IL LINGUAGGIO ASSEMBLY

Uno svantaggio non indifferente del codice macchina è la difficoltà che si incontra nella stesura e nella verifica dei programmi. Sono pochi coloro che riescono a ricordarsi tutti i codici numerici relativi alle istruzioni, ma, anche se ciò avvenisse, rimane pur sempre difficile distinguere i codici operativi (per brevità: *opcode*) dai valori usati per i dati. Non si può esaminare la parte centrale di un programma in codice macchina senza prima aver letto il programma fin dall'inizio.

Sfortunatamente, ogni microprocessore ha un proprio insieme di opcode e ciò rende difficile trascrivere un programma da un apparecchio all'altro.

Gran parte degli ostacoli vengono aggirati ricorrendo, durante la stesura di un programma, all'uso di un linguaggio intermedio, noto come *linguaggio assembly*, che si colloca a un livello appena superiore al codice macchina (e che tratteremo più a fondo in successive lezioni).

Nell'assembly, ai vari codici operativi del microprocessore corrispondono sigle, molto più facili da ricordare, chiamate anche *codici mnemonici*. Ad esempio: LD significa 'load' (caricamento), mentre JP stà per 'jump' (salto).

Con un po' di pratica, la lettura dell'assembly, anche se non del tutto immediata, risulta facile quanto quella del BASIC.

Lo svantaggio è che il computer non può leggere direttamente i programmi scritti in assembly: occorre, invece, *assemblare* il programma, impiegando un apposito *assemblatore* (che può essere anche scritto in BASIC), prima di passare all'esecuzione.

Tuttavia, questo processo risulta ancora semplice, se paragonato alla decodifica necessaria per i programmi in BASIC. Il linguaggio assembly è equivalente al codice macchina: in altre parole, ciascuna istruzione dell'assembly corrisponde esattamente a un numero nel codice macchina e lo stesso vale per ciascun carattere o cifra dei dati.

L'assemblatore, inoltre, anziché eseguire ogni istruzione durante la trasposizione, converte l'intero programma una sola volta per tutte, producendone una versione *eseguibile*.

Il Micro BBC e l'Electron sono dotati di un programma *assembler* incorporato, mentre per il Commodore, lo Spectrum e il Dragon ne sono disponibili diverse versioni commerciali. L'alternativa, per chi non possiede un assembler, è di immettere i codici manualmente. Anche la stesura dei programmi più semplici risulta facilitata se si impiega il linguaggio assembly. In un secondo momento, tavole di conversione alla mano, si passa alla trasposizione in codice macchina per un particolare modello di computer.

Può esser utile qualche prova con i programmi che seguono. In essi, il codice macchina, contenuto nelle frasi DATA, è depositato in memoria dal programma BASIC. Per comodità le frasi DATA sono raggruppate in fondo ai programmi:

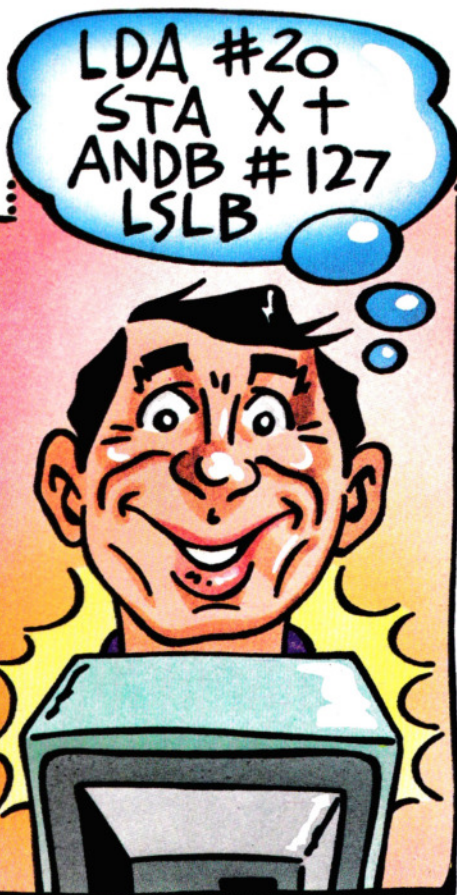
```
10 CLEAR 29999
20 FOR n=30000 TO 30020
30 READ a
40 POKE n,a
50 NEXT n
60 RANDOMIZE USR 30000
70 GOTO 60
80 DATA 17,0,88,46,0,237,95,71,58,140,92,
    128,230,63,103,1,0,3,237,176,201
```

```
10 FOR T=4047 TO 4123
20 READ A:?T=A
30 NEXT T:CALL 4047
40 DATA 160,0,185,0,128,41,7,9,144,153,0,
    124,200,169,255,153,0,124,200,192,0,208,
    235
50 DATA 174,211,15,232,224,255,208,2,162,
    128,142,211,15,174,218,15,232,224,128,
    240,8,142,218,15
60 DATA 142,224,15,208,206,162,124,142,
    218,15,142,224,15,24,173,217,15,105,1,
    41,1,141,217,15,141,223,15,76,209,15
```

```
10 CLEAR 200,31000
20 DEFUSR0=31000
30 FOR N=31000 TO 31020
40 READ A
50 POKE N,A
60 NEXT
70 N=USR0(0)
80 POKE 32767,RND(256)-1
90 FOR N=1 TO 100:NEXT
100 GOTO 70
110 DATA 142,4,0,206,136,184,166,192,184,
    127,255,138,129,167,128,140,6,0,38,242,
    57
```

```
20 FOR Z=1 TO 26
30 PRINT "■□□□□□□□□□□□
    □□□□□□□□□□□□□□□□
    □□□□□□□□□□□□□□□□";
40 NEXT Z
100 FOR Z=0 TO 25:READ X:POKE 832+Z,
    X:NEXT Z
110 SYS 832:FOR Z=1 TO 50:NEXT Z:
    GOTO 110
120 DATA 162,0,200,192,7,208,2,160,0,152,
    157,0,216,157,0,217,157,0,218
130 DATA 157,232,218,232,208,233,96
```

```
20 FOR Z=1 TO 25
30 PRINT "■□□□□□□□□□□□
    □□□□□□□□□□□□";
40 NEXT Z
100 FOR Z=0 TO 19:READ X:POKE 832+Z,
    X:NEXT Z
110 SYS 832:FOR Z=1 TO 50:NEXT Z:GOTO
    110
120 DATA 162,0,200,192,7,208,2,160,0,152,
    157,0,150
130 DATA 157,0,151,232,208,239,96
```

Adesso, si provi a scrivere un programma in BASIC, analogo a quello in codice macchina: i vantaggi di quest'ultimo risaltano immediatamente.

DIVERTIAMOCI A PROGRAMMARE LABIRINTI

I programmi di giochi, impostati su sofisticati labirinti, sono certamente complessi, ma se ne possono creare versioni più semplici con qualche ciclo FOR ... NEXT e qualche frase DATA

I labirinti hanno un fascino inesauribile sui possessori di computer e vedremo tra breve come entrare a far parte della folta schiera di creatori di giochi di questo genere, progettando i propri labirinti.

Nel nostro primo tentativo non sono previsti né ostacoli né avversari ai quali sfuggire. Tuttavia, già dai primi esempi, si cominciano a imparare le tecniche necessarie affinché gli oggetti in movimento nel labirinto non 'trapassino' i muri che delimitano il percorso obbligato, una delle necessità basilari di tutti i labirinti. Comunque, per rendere questo primo gioco più competitivo, è previsto un conteggio di punti e dei tempi, un *totalizzatore* del punteggio massimo raggiunto.

Il miglior sistema è sempre quello di partire dalla realizzazione degli elementi di base, sviluppando gradualmente su di essi il resto del gioco. Ecco come definire l'immagine di un labirinto sullo Spectrum:

```
100 FOR n=3 TO 17
110 READ a$
120 FOR m=7 TO 21
130 PRINT AT n,m;"."
140 IF a$(m-6)="p" THEN PRINT PAPER
    1; INK 1;AT n,m;"□"
150 NEXT m
160 NEXT n
9000 DATA "ppppppppppppppp"
9010 DATA "p.............p"
9020 DATA "p.pp.pp.pp.pp.p"
9030 DATA "p.p.........p.p"
9040 DATA "p...p.p.p.p...p"
9050 DATA "p.ppp.p.p.ppp.p"
9060 DATA "p.....p.p.....p"
9070 DATA "pppp.pp.pp.pppp"
9080 DATA "p.....p.p.....p"
9090 DATA "p.ppp.p.p.ppp.p"
9100 DATA "p...p.p.p.p...p"
9110 DATA "p.p.........p.p"
9120 DATA "p.pp.pp.pp.pp.p"
9130 DATA "p.............p"
9140 DATA "ppppppppppppppp"
```

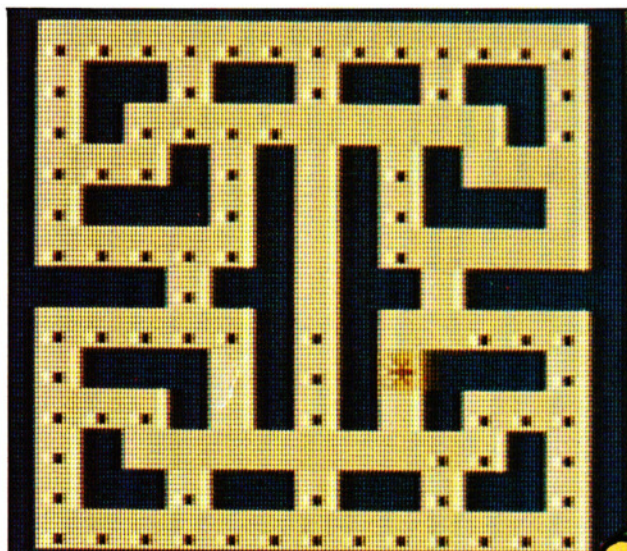
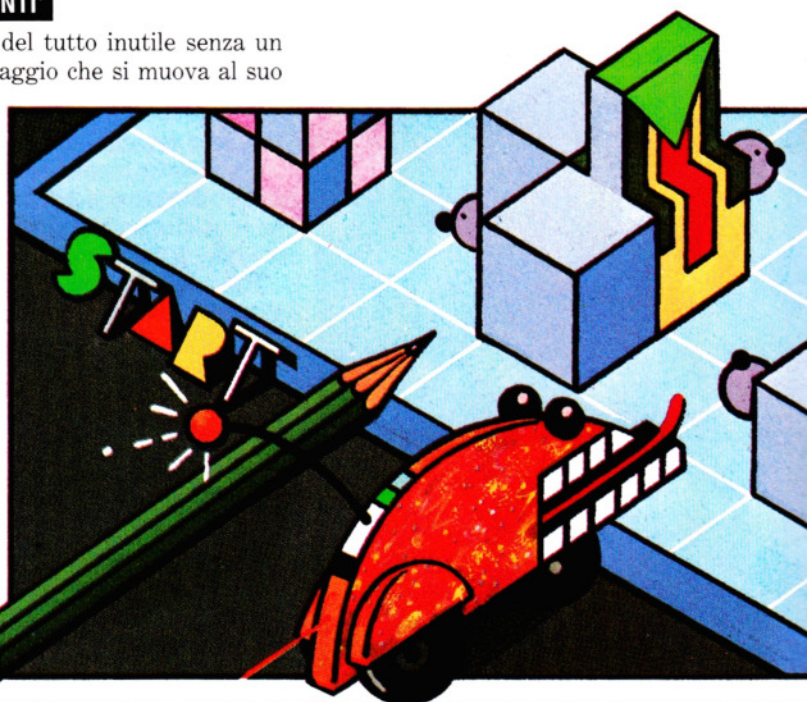
Le linee 100, 120, 150 e 160, con un paio degli ormai familiari cicli FOR ... NEXT, costruiscono i confini del labirinto. La linea 130 visualizza, mediante una PRINT, un punto in ciascun quadrato del percorso.

A questo punto, il controllo passa alle linee 110 e 140, che leggono i valori contenuti nelle frasi DATA (linee da 9000 a 9140), convertendo ciascun punto in uno spazio, ma in colore blu su sfondo blu, se il dato letto è la lettera p. Sullo Spectrum non si possono sovrapporre due caratteri nella medesima locazione dello schermo.

IL 'MANGIAPUNTI'

Un labirinto è del tutto inutile senza un qualche personaggio che si muova al suo interno. Perciò aggiungiamo:

```
50 LET x=10
60 LET y=14
1000 PRINT PAPER 6; INK 2;AT x,y;"*"
1010 LET xx=x
1020 LET yy=y
1030 IF INKEY$="" THEN GOTO 1030
1040 IF INKEY$="p" AND ATTR (x-1,y)<
    >9 THEN LET x=x-1
1050 IF INKEY$="l" AND ATTR (x+1,y)<
    >9 THEN LET x=x+1
1060 IF INKEY$="z" AND ATTR (x,y-1)<
    >9 THEN LET y=y-1
1070 IF INKEY$="x" AND ATTR (x,y+1)<
    >9 THEN LET y=y+1
1080 PRINT INK 7;AT xx,yy;"□"
1090 GOTO 1000
```

Se si è già fatta pratica con il controllo dei movimenti da tastiera (vedere alle pagine 54-59), la maggior parte di queste linee di programma sarà già nota. Il nostro 'amico' asterisco è visualizzato nella posizione x,y e viene fatto muovere dal giocatore mediante una serie di INKEY$. (Un particolare importante: il personaggio entra soltanto nei quadrati che *non* sono blu, vale a dire se ATTR *non* vale 9, il numero iden-

1. Un semplice labirinto da creare sullo Spectrum

tificativo del colore usato per tracciare i contorni del labirinto).

Volendo un labirinto di colore diverso, il valore di ATTR si calcola così:

1 Si prende il codice di colore, riportato sulla tastiera, usato per la INK. Nel nostro caso questo vale 1.
2 Si prende il codice di colore usato per PAPER (1 nel nostro caso) e lo si moltiplica per 8.
3 Si somma 64 se nell'area in questione è usato l'attributo BRIGHT. Nel nostro caso, il valore è 0.
4 Si aggiunge 128 se l'area in questione è in FLASH. Nel nostro caso, il valore è 0.

Adesso si sommano tutti i valori: il risultato è il numero da dare a ATTR.

La variabile ATTR può venir impiegata anche per altri scopi, oltre a quello di evitare che il personaggio *sconfini*. Per esempio, può servire per indicare se, essendo il personaggio entrato in una zona 'proibita', è il caso di farlo *esplodere*. Nel caso di un INK rosso e di un PAPER rosso, sarebbe necessaria una linea del tipo:

```
IF ATTR (x,y) = 18 THEN ...
```

Nel frattempo, le linee 1010, 1020 e 1080 definiscono la posizione appena lasciata dal personaggio, provvedendo a cancellare (con la PRINT di uno spazio) il puntino 'mangiato'. Si noti la loro collocazione: subito prima e subito dopo le INKEY$.

La linea 1090 è provvisoria. Il suo scopo è di permettere una prima verifica del programma. Senza il ciclo da essa introdotto, infatti, il personaggio si muoverebbe di un solo quadrato e subito dopo il programma terminerebbe.

PUNTEGGI E TEMPI

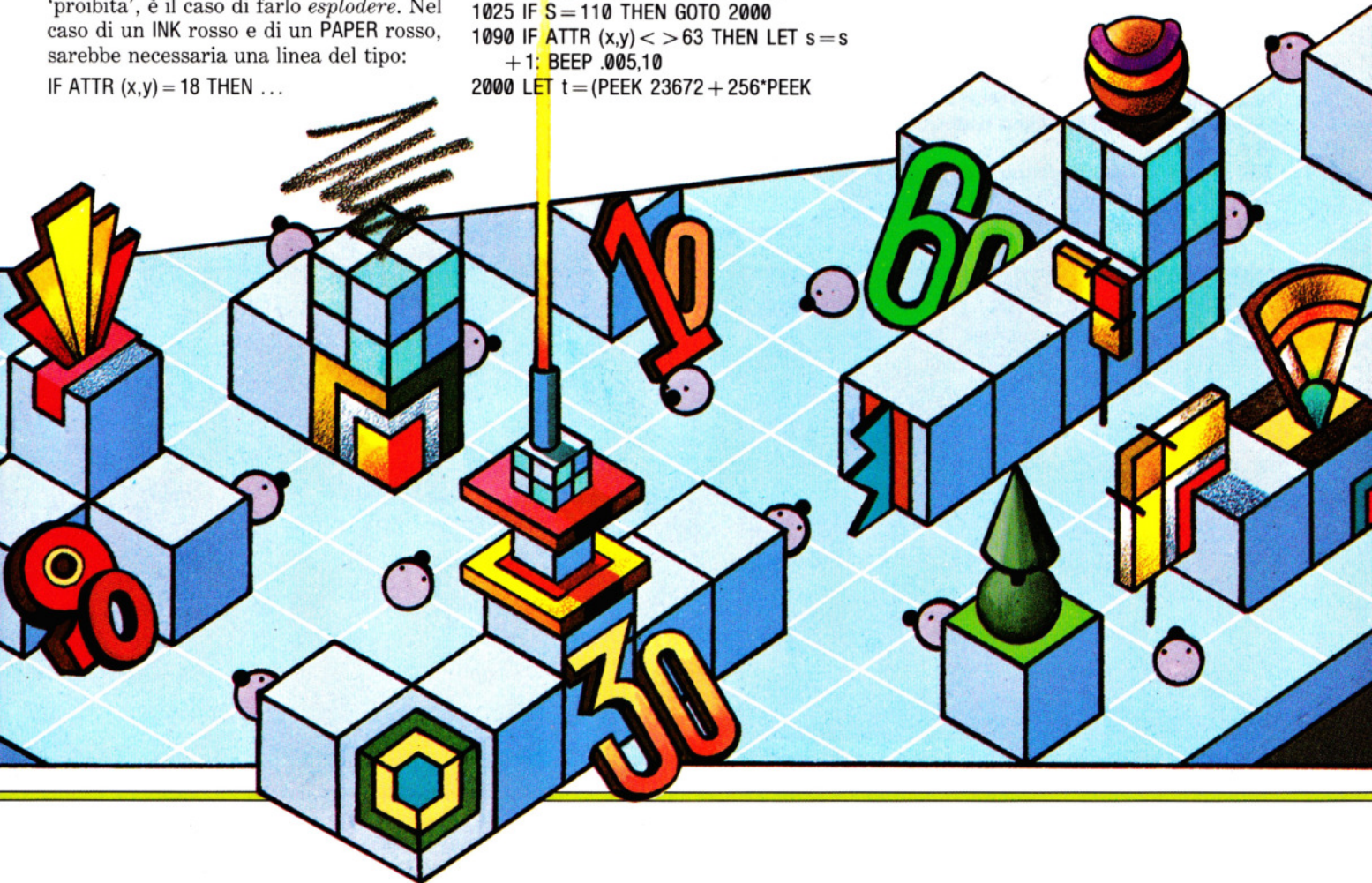
Adesso abbiamo l'occorrente per un semplice gioco. Per renderlo più interessante, possiamo dotarlo di una routine per il calcolo del punteggio e dei tempi. Si aggiungano le seguenti linee:

```
10 LET bt=100000
40 LET s=0
990 POKE 23672, 0: POKE 23673, 0
1025 IF S=110 THEN GOTO 2000
1090 IF ATTR (x,y)< >63 THEN LET s=s
    +1: BEEP .005,10
2000 LET t=(PEEK 23672+256*PEEK
    23673)/50
2010 PRINT AT 1,6;t;"□SECONDI□"
2020 IF t<bt THEN LET bt=t
2030 PRINT AT 19,4;"MIGLIOR TEMPO□";
    bt;"□SECONDI"
```

Per verificare il funzionamento, serve una linea provvisoria, da cancellare in seguito:

```
1100 GOTO 1000
```

La parte relativa al punteggio è semplice. La linea 40 azzera i punti. La linea 1090 somma 1 al punteggio ogni volta che viene mangiato un puntino o, per meglio dire, quando INK e PAPER del quadrato occupato non sono ambedue bianchi (altro uso di ATTR!). La linea 1025, che entra in azione se tutti i puntini sono stati mangiati, consente la verifica del tempo impiegato dal giocatore (linea 2000).

Il calcolo dei tempi è leggermente più

complesso, ricorrendo a delle PEEK e POKE, la cui funzione verrà spiegata oltre. Ma, in breve, la linea 99Ø azzera l'orologio dello Spectrum depositando uno Ø in una particolare locazione di memoria. La linea 2ØØØ conta il numero di quadri comparsi sullo schermo dall'inizio del gioco e li divide per 5Ø, ottenendo il tempo (approssimativo) in secondi. Il 'miglior tempo' viene inizialmente fissato dalla linea 1Ø in 1ØØ.ØØØ, un valore altissimo, che verrà sicuramente battuto da qualsiasi giocatore. La linea 2Ø2Ø confronta questo con il tempo ottenuto nel gioco.

UN ALTRO GIRO?

Per dare al giocatore la possibilità di riprovare, servono queste poche linee, ma prima occorre digitare CAPS SHIFT e BREAK, poi ENTER:

```
2Ø4Ø PRINT AT 21,2;"PREMERE QUALSIASI
   TASTO PER GIOCARE ANCORA"
2Ø5Ø IF INKEY$<>"" THEN GOTO 2Ø5Ø
2Ø6Ø IF INKEY$="" THEN GOTO 2Ø6Ø
2Ø7Ø RESTORE
2Ø8Ø GOTO 4Ø
```

La RESTORE serve per riutilizzare le frasi DATA più volte.

ALTRI LABIRINTI

Per creare altri labirinti, con le stesse dimensioni di questo, basta cambiare la sequenza dei punti e delle p nelle frasi DATA (dalla linea 9ØØØ in poi). Per un labirinto più grande o più piccolo, occorre ridefinire i contorni, agendo sui valori nelle linee 1ØØ e 12Ø.

Nel far ciò, è bene assicurarsi di avere sufficienti elementi nelle frasi DATA, altrimenti si ottiene un messaggio d'errore.

Il programma per gli Acorn è suddiviso in tre segmenti. Il primo traccia il labirinto sullo schermo, il secondo provoca il movimento del personaggio al suo interno e il terzo, infine, provvede a calcolare e a visualizzare tempi e punteggi.

Ecco la prima parte del programma.

Eseguendola, a sinistra sullo schermo compare un labirinto verde, il cui percorso è costellato da una serie di punti gialli.

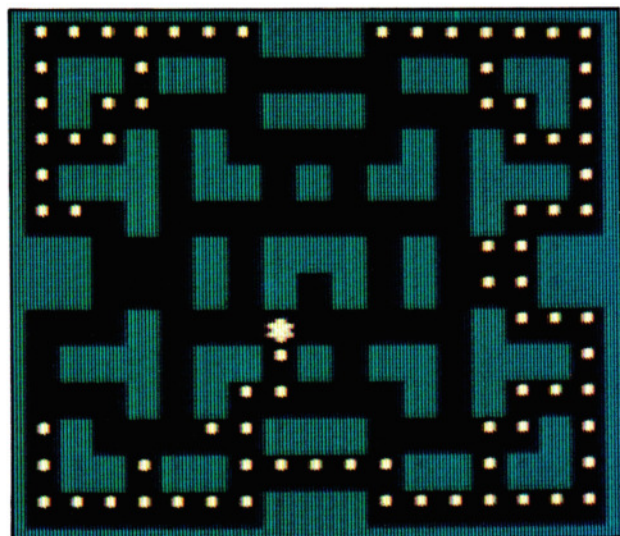
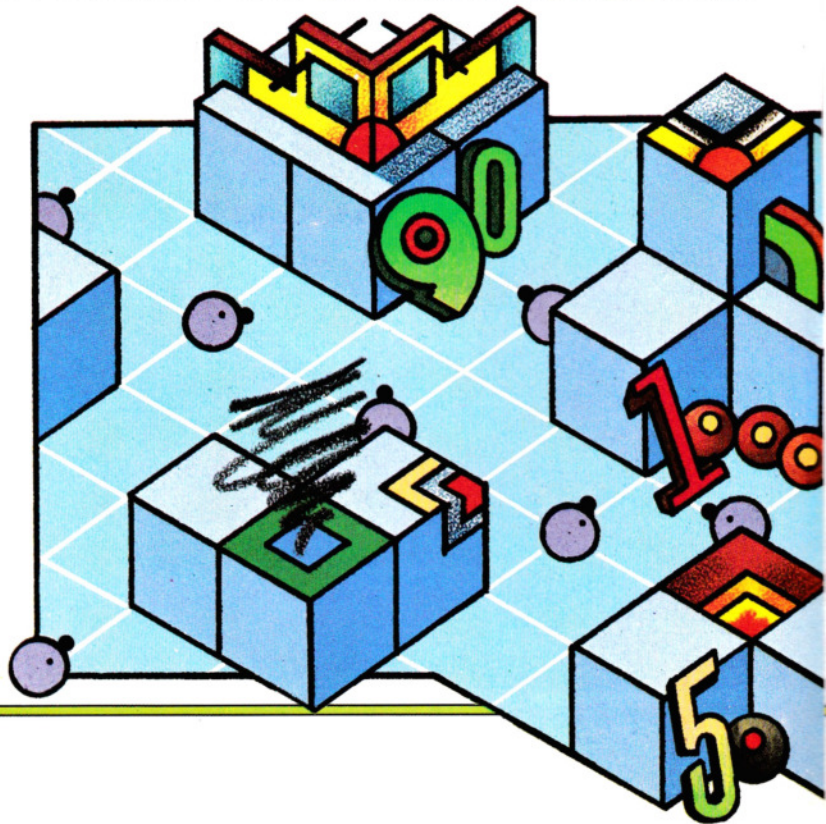
```
1Ø MODE 1
3Ø VDU 23,224,255,255,255,255,255,255,
   255,255
4Ø VDU 19,1,4;Ø;19,3,5;Ø;
5Ø CLS
8Ø VDU 5
85 □
11Ø FOR riga=8 TO 23
12Ø READ A$
13Ø FOR colonna=1 TO 19
14Ø MOVE colonna*32, riga*32
15Ø IF MID$(A$,colonna,1)="A" THEN GCOL
   Ø,1:PRINT CHR$224 ELSE GCOL Ø,2:
   PRINT "."
16Ø NEXT colonna
17Ø NEXT riga
46Ø □
5ØØ DATA AAAAAAAAAAAAAAAAAAA
51Ø DATA A.......AAA.......A
52Ø DATA A.AA.AA.....AA.AA.A
53Ø DATA A.A.....AAA.....A.A
54Ø DATA A...A.A.....A.A...A
55Ø DATA A.AAA.AA.A.AA.AAA.A
56Ø DATA A...A.........A...A
57Ø DATA AAA...A.AAA.A...AAA
58Ø DATA AAA...A.AAA.A...AAA
59Ø DATA A...A.........A...A
6ØØ DATA A.AAA.AA.A.AA.AAA.A
61Ø DATA A...A.A.....A.A...A
62Ø DATA A.A.....AAA.....A.A
63Ø DATA A.AA.AA.....AA.AA.A
64Ø DATA A.......AAA.......A
65Ø DATA AAAAAAAAAAAAAAAAAAA
```

La parte principale di questo listato è l'elenco delle frasi DATA: ciascuna linea corrisponde a una riga orizzontale del labirinto, definendone la forma (le A rappresentano i muri). Questo metodo di raggruppare le frasi DATA è molto conveniente, poiché basta un'occhiata al listato per intravvedere già la forma del labirinto e apportare, nel caso, delle modifiche.

Il resto del programma converte ogni A in un quadrato, visualizzandolo nella giusta posizione sullo schermo. Sembra facile... ma non mancano i problemi.

COME FUNZIONA

Prima di tutto occorre definire i quadrati, poiché questi non esistono già pronti. Il modo più semplice è di definire un UDG mediante una VDU 23 come si vede alla linea 3Ø. Il successivo problema è che, nel Modo 1, i colori disponibili sono nero, rosso, giallo e bianco, mentre vogliamo un labirinto verde. La linea 4Ø usa una VDU 19 per ridefinire i colori, in modo da sostituire il blu al rosso (la manipolazione dei colori sarà oggetto di una lezione a parte).

L'ultima complicazione è la VDU 5, alla linea 5Ø. Questa serve per usare il cursore grafico ai fini della visualizzazione, permettendo l'impiego delle coordinate grafiche. In altre parole, al posto di PRINT

2. La versione Dragon mostra anche il vostro tempo

TAB (colonna, riga), vengono usate delle MOVE colonna*32, riga*32, seguite da delle PRINT. La moltiplicazione per 32 serve a convertire le colonne e le righe in coordinate grafiche. Nel Modo 1, ad esempio, si hanno 40 colonne: moltiplicando 40×32, si ottiene 1280, la larghezza dello schermo grafico. Ora, il programma può passare al tracciamento del labirinto nella parte sinistra dello schermo, tra le righe 8 e 23 e le colonne da 1 a 19.

La linea 110 apre un ciclo nel quale la linea 120 legge la frase DATA per ogni riga del labirinto, chiamandola A$. La linea 130 apre un ciclo sulle colonne di ogni riga e la linea 140 posiziona il cursore. La linea 150 fa il resto del lavoro: se il carattere è una A, visualizza un blocco blu, altrimenti un punto giallo. La funzione MID$ serve per scandire gli elementi DATA di una posizione alla volta.

MUOVERSI NEL LABIRINTO

Fin qui, il programma non ha fatto altro che visualizzare il labirinto. Aggiungiamo queste poche linee per muovere l'asterisco 'mangiapunti':

```
60 *FX11,15
70 *FX12,15
190 LET X=10*32: LET Y=16*32
200 MOVE X,Y: GCOL 0,0: VDU 224,8: GCOL
    0,3: PRINT "*"
220 LET LX=X: LET LY=Y
230 LET K$=GET$
240 IF K$="Z" AND POINT(X-32,Y)<>1
    THEN X=X-32
250 IF K$="X"AND POINT(X+32,Y)<>1
    THEN X=X+32
260 IF K$="L" AND POINT(X,Y-32)<>1
    THEN Y=Y-32
270 IF K$="P" AND POINT(X,Y+32)<>1
    THEN Y=Y+32
290 MOVE LX,LY:GCOL0,0:VDU224
300 GOTO 200
```

Adesso, diamo un RUN al programma. Z, X, P e L, al solito, muovono l'asterisco.

Queste linee assomigliano molto alla routine per muovere un carattere, già presentata alle pagine 11 e 12, ma stavolta il programma deve evitare che l'asterisco attraversi i muri del labirinto. Questo compito spetta alla funzione POINT.

POINT (X,Y) controlla il colore *grafico* al punto X,Y. (Non è possibile controllare il colore dei testi, non esistendo una simile funzione nel computer: ecco perché è stato adoperato il cursore grafico. Questo sistema permette di visualizzare i blocchi e i puntini, associando loro i colori desiderati, il cui codice è verificabile).

Il funzionamento di POINT, in questo programma, è il seguente: quando viene premuto un tasto, POINT controlla il codice del colore associato al quadrato verso il quale ci si muove. Se questo vale 1 (=colore blu), allora il movimento è ignorato, altrimenti si procede.

Qui incontriamo di nuovo il numero 32. Si ricordi che un salto di 32 corrisponde al movimento di una posizione del carattere.

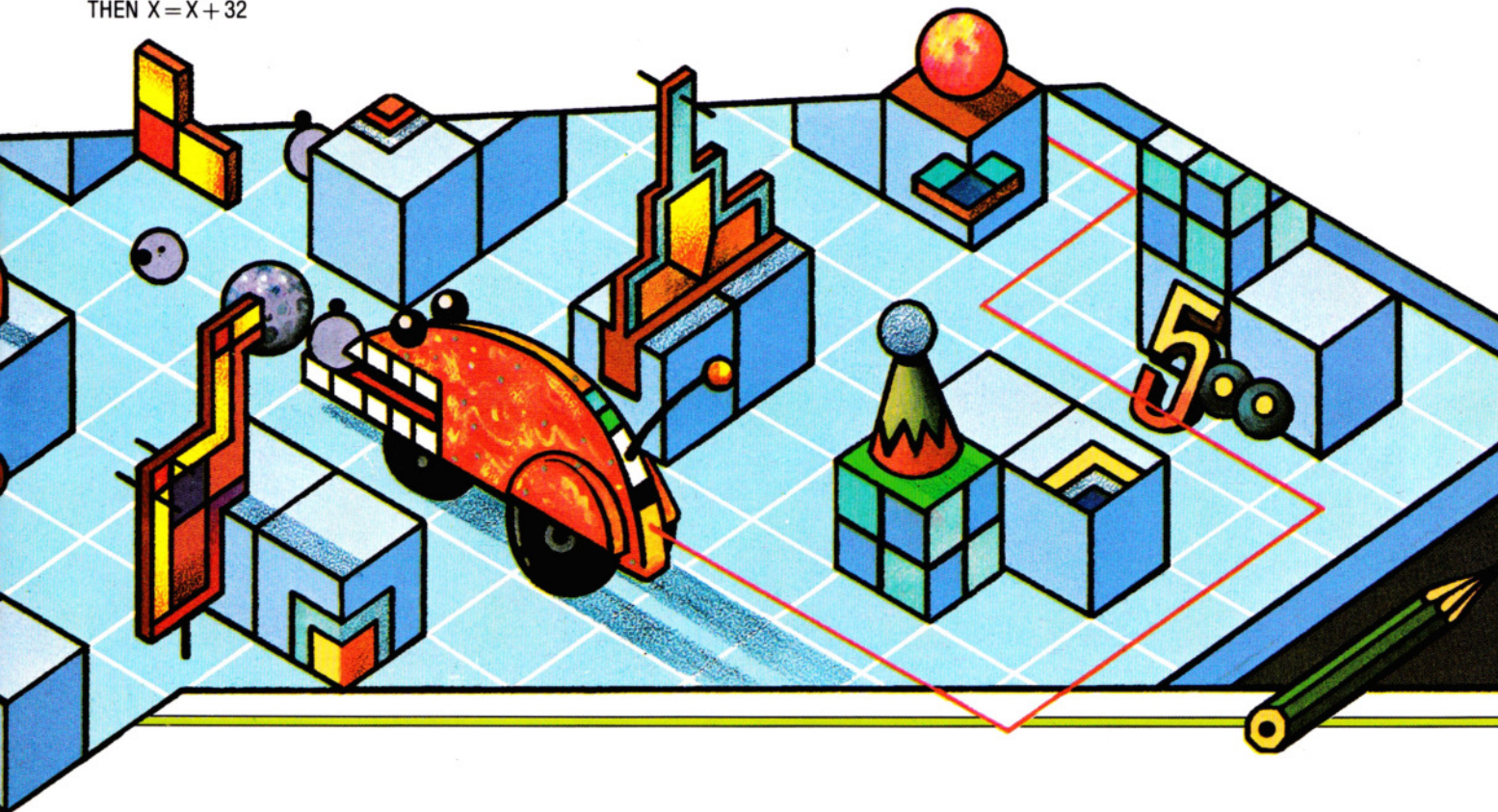
Per quanto concerne le restanti linee, la 190 assegna all'asterisco la sua posizione iniziale: la 200 vi sposta l'asterisco, provvedendo anche a far sparire (con una VDU 224) l'eventuale punto che si trovasse in tale posizione. VDU 224 equivale a PRINT CHR$ (224), ma il suo uso è più conveniente. L'altro numero, ossia 8, fa indietreggiare di una posizione il cursore, per visualizzare correttamente l'asterisco. La linea 290 *pulisce* la posizione precedente.

Eseguendo il programma, si nota che l'asterisco si muove lentamente, esitando quando si tratta di mutare direzione. Le linee 60 e 70 ovviano a questo inconveniente, agendo sulla ripetizione automatica dei tasti. Infatti, *FX11,15 e *FX12,15, portano a 15 centesimi di secondo, rispettivamente, la pausa d'attesa e la frequenza di *autorepeat* dei tasti.

Rimane un problema: quando si esce dal programma con ESCAPE, le funzioni *FX rimangono attive, ostacolando notevolmente l'immissione di successivi comandi: LIST, ad esempio, può diventare LLIISTT! Il rimedio è semplice, perché basta premere BREAK, poi digitare OLD, e nuovamente LIST. Il tasto BREAK disattiva le funzioni *FX.

CRONOMETRARE IL GIOCO

Il programma non è ancora completo, senza un conteggio del tempo impiegato a 'mangiare' i puntini. Ma bastano poche linee per porre rimedio a questa mancanza:

```
20 LET tmigliore=999999
90 LET punti=0
100 RESTORE
180 TI=0
210 IF punti=154 THEN GOTO 310
280 IF POINT(X+12,Y-20)=2 THEN punti
    =punti+1:SOUND 0,-15,1,1
```

```
310 LET ti=TI/100
320 IF tmigliore>ti THEN tmigliore=ti
330 MOVE X,Y:GCOL0,0:VDU 224
340 *FX12,0
350 VDU 4
360 VDU 23;8202;0;0;0;
370 FOR ritardo=1 TO 1000:NEXT
380 COLOUR3
390 PRINT TAB(24,10) "TEMPO□="
400 PRINT TAB(24,11);ti;"□sec."
410 PRINT TAB(24,13) "MIGLIOR TEMPO"
420 PRINT TAB(24,14);tmigliore;"□sec."
430 PRINT TAB(24,20) "PREMERE RETURN"
440 PRINT TAB(24,21) "PER GIOCARE
   ANCORA"
450 IF GET=13 THEN GOTO 50 ELSE GOTO
   450
```

Inizialmente, il 'miglior tempo' è fissato in 999.999 secondi e il numero di puntini viene azzerato. La linea 1ØØ è indispensabile per poter rileggere le frasi DATA a ogni nuovo gioco. Senza di essa, otterremmo un irritante messaggio OUT OF DATA al secondo giro.

La linea 28Ø controlla se ci si muove 'sopra' a un puntino. Viene nuovamente impiegata la funzione POINT: se questa rileva che il colore del quadrato vale 2 (=giallo, nel nostro caso), allora il punteggio aumenta di 1 e viene emesso un suono appropriato. Se, al contrario, l'asterisco si sposta in un quadrato vuoto, la linea viene ignorata. La figura 3 mostra una fase intermedia di un gioco.

Esauriti tutti i puntini, l'esecuzione, grazie alla linea 21Ø, passa alla linea 31Ø, che blocca il cronometro e divide il valore di TIME per 1ØØ, ottenendo i secondi trascorsi. Questo valore viene confrontato con il miglior tempo (linea 32Ø), sostituendosi, eventualmente, ad esso. La linea 33Ø cancella l'asterisco. Le due successive linee servono per riportare il computer alle condizioni normali. La *FX12,Ø disattiva quanto impostato dalle varie *FX, mentre VDU 4 disattiva VDU 5.

Infine, a destra sullo schermo, vengono visualizzati il tempo del gioco e il miglior tempo. La linea 45Ø attende un carattere e, se questo corrisponde a RETURN, il gioco ricomincia. Per uscire dal programma occorre battere il tasto BREAK.

DISEGNARE IL PROPRIO LABIRINTO

Il primo passo consiste nel disegnare su carta la forma voluta. Poi, usando una A per ogni sezione di muro e un punto per le altre, si codificano le frasi DATA per il proprio labirinto.

Se questo ha dimensioni maggiori o minori di quello illustrato nel nostro esempio, occorre apportare qualche modifica anche alle linee di programma 11Ø e 13Ø, dove vengono predisposti i cicli FOR ... NEXT che leggono le frasi DATA.

Quasi sicuramente, il numero di puntini del nuovo labirinto sarà diverso: di conseguenza, cambiare la linea 21Ø.

Queste modifiche dovrebbero bastare, salvo che le dimensioni del labirinto siano tali da dover spostare le scritte relative ai tempi. In tal caso, occorre procedere per tentativi, aggiustando la posizione delle varie PRINT, fino al risultato desiderato.

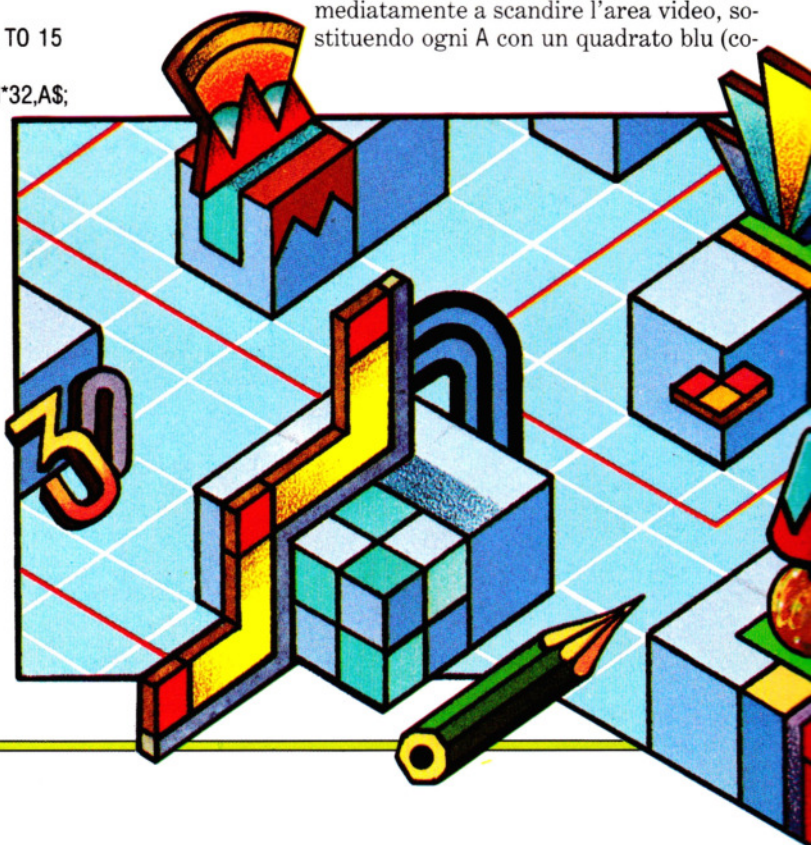
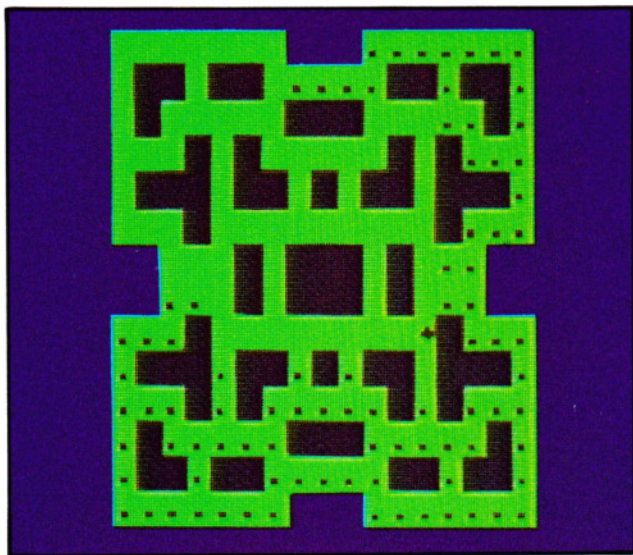
Il labirinto è così definito:

```
30 CLS4
40 LET P=297
1000 FOR N=0 TO 15
1010 READ A$
1020 PRINT@ N*32,A$;
1030 FOR M=0 TO 18
1040 IF PEEK(1024+N*32+M)=65 THEN
   POKE(1024+N*32+M),175
1050 NEXT M
1060 NEXT N
2000 DATA AAAAAAAAAAAAAAAAAAA
2010 DATA A.......AAA.......A
2020 DATA A.AA.AA.....AA.AA.A
2030 DATA A.A.....AAA.....A.A
2040 DATA A...A.A.....A.A...A
2050 DATA A.AAA.AA.A.AA.AAA.A
2060 DATA A...A.........A...A
2070 DATA AAA...A.AAA.A...AAA
2080 DATA AAA...A.AAA.A...AAA
2090 DATA A...A.........A...A
2100 DATA A.AAA.AA.A.AA.AAA.A
2110 DATA A...A.A.....A.A...A
2120 DATA A.A.....AAA.....A.A
2130 DATA A.AA.AA.....AA.AA.A
2140 DATA A.......AAA.......A
2150 DATA AAAAAAAAAAAAAAAAAAA
```

La forma del labirinto dipende dai valori contenuti nelle DATA, linee da 2ØØØ a 215Ø, dove le A individuano i contorni.

Prima della lettura delle DATA, la linea 3Ø ripulisce lo schermo e lo colora in rosso, il cui codice è 4. La linea 4Ø assegna all'asterisco la posizione di partenza.

Le linee da 1ØØØ a 1Ø6Ø si occupano di tracciare il labirinto sullo schermo. A ogni iterazione del ciclo FOR ... NEXT, il programma legge una nuova frase DATA (linea 1Ø1Ø), chiamandola A$; la linea 1Ø2Ø ne visualizza il contenuto sullo schermo.

Ma una sequenza di puntini e di A sullo schermo non rappresenta certo un labirinto, cosicché la linea 1Ø4Ø provvede immediatamente a scandire l'area video, sostituendo ogni A con un quadrato blu (co-

3. Un labirinto semplice versione Electron o BBCB

dice 175). Il ciclo nelle linee 1Ø3Ø e 1Ø5Ø assicura che la linea 1Ø4Ø non trascuri alcuna locazione dell'area video, durante il lavoro di conversione dei codici.

Per ottenere il movimento di un asterisco (il nostro personaggio) nel labirinto, si aggiungano le seguenti linee. (Nel Tandy cambiare i 223 in 247 nelle linee 11ØØ e 111Ø; sostituire 239 con 251 nella linea 112Ø e modificare 247 in 253 nella 113Ø).

```
1080 PRINT @ P,"*";
1090 LET LP=P
1100 IF PEEK(340)=223 AND PEEK(1023
    +P)<>175 THEN LET P=P-1
1110 IF PEEK(338)=223 AND PEEK(1025
    +P)<>175 THEN LET P=P+1
1120 IF PEEK(338)=239 AND PEEK(992
    +P)<>175 THEN LET P=P-32
1130 IF PEEK(342)=247 AND PEEK(1056
    +P)<>175 THEN LET P=P+32
1150 PRINT @ LP,"□";
1170 GOTO 1080
```

Questa sezione di programma è simile a quella presentata a pagina 13, ma esistono delle differenze: l'asterisco, ad esempio, non tenta di attraversare i muri del labirinto, mentre la linea 11ØØ controlla due locazioni di memoria: quella che corrisponde al tasto Z e quella che si riferisce alla posizione nella quale sta per avventurarsi l'asterisco. Qualora si stia premendo il tasto Z (codice 223), e, *al tempo stesso*, la locazione video *non* contenga un quadrato blu (codice 175), allora il movimento è consentito.

I tasti di controllo del movimento sono identici alla precedente versione (Z = sinistra, X = destra, P = su, L = giù). Si provi a muovere l'asterisco nel labirinto, in modo da 'mangiare' i puntini.

QUANTO SIAMO VELOCI?

Con l'aggiunta di poche linee, possiamo misurare il tempo necessario per eliminare tutti i puntini:

```
20 LET FA=999999
50 RESTORE
60 LET D=0
1070 TIMER=0
1140 IF PEEK(1024+P)=110 THEN PLAY
    "T80B":LET D=D+1
1160 IF D=153 THEN GOTO 1180
1180 PRINT@ P,"□";
1190 TI=TIMER/50
1200 PRINT@52,"TEMPO=";
1210 PRINT@84,TI;"SEC□";
1220 IF FA>TI THEN FA=TI
1230 PRINT@212,"MIGLIOR TEMPO";
1240 PRINT@244,FA;"SEC.□";
1250 PRINT@340,"PREMERE";
1260 PRINT@372,"ENTER";
1270 PRINT@404,"PER";
1280 PRINT@436,"RICOMINCIARE";
1290 LET IN$=INKEY$:IF IN$="" THEN
    GOTO 1290
1300 IF IN$=CHR$(13) THEN GOTO 40
1310 GOTO 1290
```

Quando si esegue il programma, alla destra del labirinto appare il tempo, assieme al 'miglior tempo'. Quest'ultimo valore viene inizialmente posto uguale a 999.999 secondi nella linea 2Ø. La linea 6Ø azzera il punteggio, mentre la 1Ø7Ø azzera il cronometro.

La linea 114Ø esamina la locazione video nella quale si trova l'asterisco, per vedere se contiene un puntino. In caso affermativo, il comando PLAY emette un appagante suono di avvenuta... digestione! Contemporaneamente, viene incrementato il punteggio (variabile D)

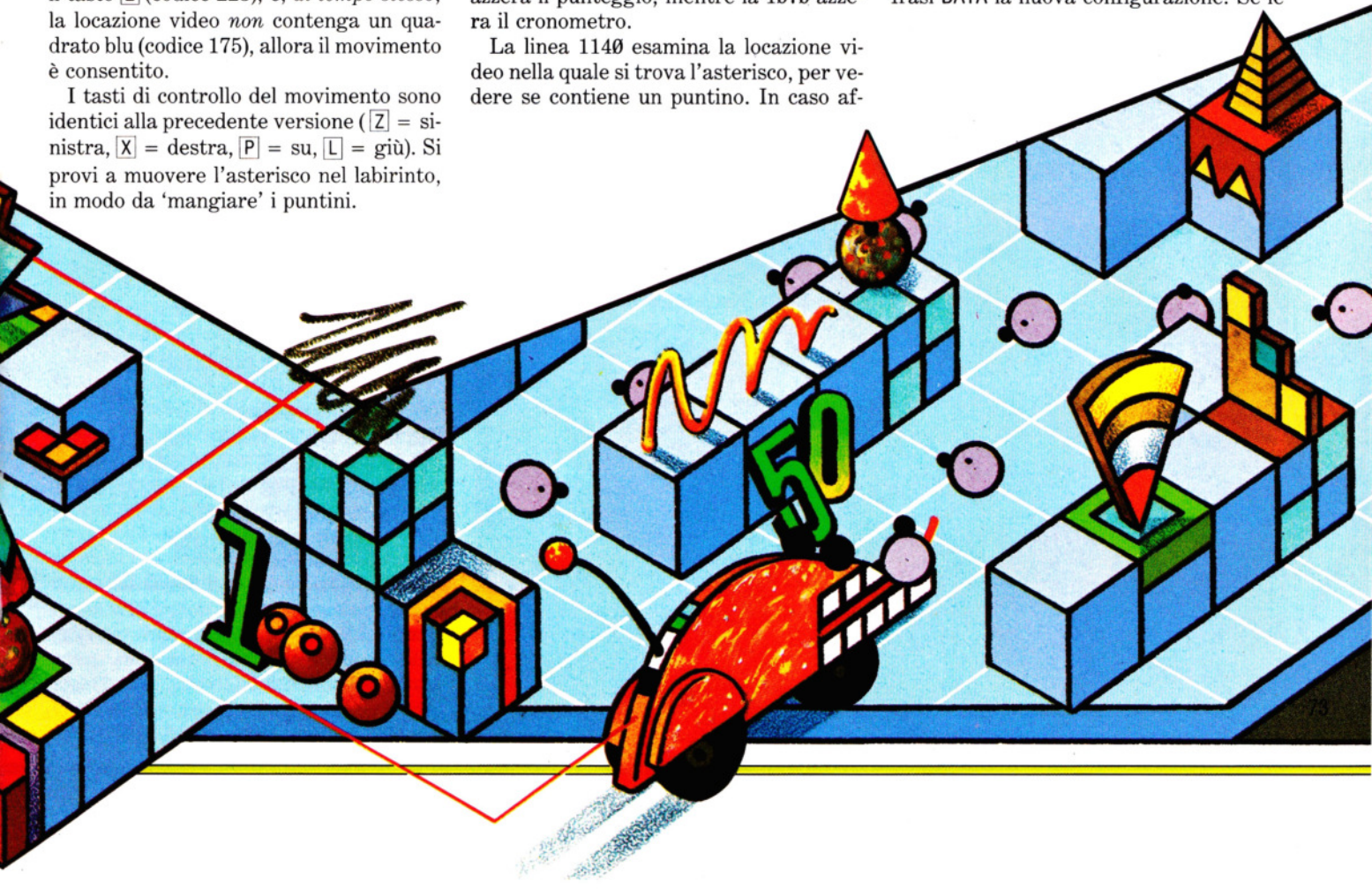
La linea 116Ø controlla se i puntini sono esauriti, saltando, in tal caso, alla linea 118Ø, che cancella l'asterisco.

Alla linea 119Ø, TI=TIMER/5Ø blocca il cronometro, convertendo il tutto in secondi. Le linee 12ØØ e 121Ø visualizzano i tempi sullo schermo.

La linea 122Ø verifica se si è battuto il 'miglior tempo' (FA) con quello appena ottenuto (TI). In caso affermativo, TI diventa il nuovo 'miglior tempo'. Le linee 123Ø e 124Ø visualizzano il miglior tempo, prima di invitare a un nuovo 'giro', che si ottiene premendo il tasto ENTER.

La linea 129Ø attende la pressione di un tasto e la 13ØØ verifica che questo corrisponda a ENTER. Se questo è il caso, l'esecuzione riparte dalla RESTORE alla linea 5Ø, che consente di rileggere nuovamente le frasi DATA. Per uscire dal programma, occorre premere il tasto BREAK.

IL PROGETTO DI UN LABIRINTO

Conviene adoperare un foglio di carta quadrettata per tracciare un nuovo labirinto, ricordandosi che l'area di schermo disponibile misura 32×16 caratteri. Progettato il labirinto, basta riportare nelle frasi DATA la nuova configurazione. Se le

dimensioni sono diverse da quelle dell'esempio, allora devono essere modificate le linee 1000 e 1030 (cicli FOR ... NEXT): si contano quante righe occupa il nuovo labirinto e si sottrae 1. Il risultato va inserito alla fine della linea 1000. Analogamente, si calcola il numero di colonne, aggiustando il valore nella linea 1030.

Infine, si conta il numero di punti contenuti nel nuovo labirinto: se questo è diverso da 153, occorre sostituire questo valore, contenuto nella linea 1160.

Il labirinto è ottenuto con delle PRINT di caratteri grafici presenti nelle ROM.

Lanciando il programma, in posizione centrata sullo schermo appare il labirinto. Il movimento del carattere π si ottiene coi tasti Z (sinistra), X (destra), P (sù) e L (giù). Il simbolo π, al suo passaggio, mangia i punti (o monetine), incrementando un apposito contatore.

In questa forma, il programma può non essere appassionante, tuttavia mostra con chiarezza come definire un labirinto. Osservando il listato, si nota come le linee da 10 a 40 servono per definire i valori iniziali: P1 è la locazione di memoria video dalla quale iniziano le monetine; BD è l'indirizzo del contorno, mentre BG è quello del colore dello sfondo. A questi indirizzi vengono depositati, rispettivamente, i valori corrispondenti al blu e al nero, nelle linee 45 e 50. Il colore dei caratteri è fissato nella linea 30: alterando il valore tra parentesi (purché con uno dei valori consentiti e riportati nel manuale), si cambia colore al labirinto.

La linea 35 mostra un diverso modo di ottenere CLR/HOME: precedentemente abbiamo usato il carattere 'cuore inverso', per pulire lo schermo e posizionare il cursore in alto a sinistra. Lo stesso risultato si ha con CHR$(147), ma conviene definire tale valore come variabile 'stringa', all'inizio di un programma, come accade nel nostro esempio, per non rallentare l'esecuzione.

La linea 55 predispone la ripetizione automatica dei tasti (autorepeat), essenziale in un gioco di questo genere.

Poi viene la sequenza che visualizza il labirinto, seguita, alla linea 200, da una POKE, che colloca il simbolo nella locazione 1117, l'angolo in alto a sinistra del labirinto. Viene quindi richiesta la pressione di un tasto per iniziare (o continuare) il gioco. La PRINT alla linea 225 contiene due caratteri 'cursore in alto' e quaranta spazi. Non appena viene premuto un tasto 'spazio', questa PRINT provoca l'eliminazione della scritta: un facile e comodo metodo di cancellazione. Lo stesso risultato si ha con:

```
225 PRINT "◘◘◘":FOR T=0 TO 39:
   PRINT "□";:NEXT T
```

che, usando un ciclo FOR ... NEXT, è una formulazione più *elegante*.

Le successive linee da 300 a 320 attendono e controllano la pressione di un tasto, usando una GET e una serie di IF ... THEN per modificare P2, cioè la nuova posizione del simbolo π. La linea 325 controlla se il movimento è consentito, saltando alla 350 in caso negativo. Se, invece, la locazione contiene una monetina, la linea 330 incrementa il conteggio dei punti, la 335 pulisce la posizione precedente P1 e la linea 340 attribuisce un nuovo valore alla posizione. Il programma prosegue con la visualizzazione del contatore di monete e l'esecuzione torna alla linea 300, salvo quando il contatore di monete raggiunge il valore 103 (linea 365).

Per usare il programma sul Vic 20, occorrono diverse modifiche: omettere le linee 15 e 45; nella linea 10 usare 7735 invece di 1117; nella linea 20 usare 36879 invece di 53281. Nella linea 25 usare 22 invece di 40. Nella linea 50 usare 14 e non 0. Nelle linee da 100 a 170 usare TAB(3) e non TAB(12). Modificare il numero di spazi nella linea 210 per una corretta visualizzazione, aggiungendone altrettanti nella linea 225. Nella linea 360 cambiare TAB(14) con TAB(4).

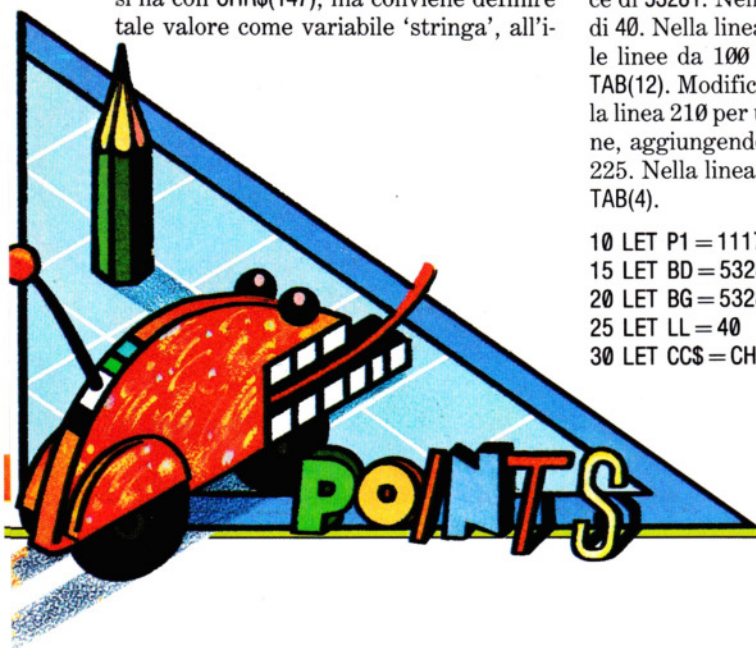
```
10 LET P1=1117: LET P2=P1
15 LET BD=53280
20 LET BG=53281
25 LET LL=40
30 LET CC$=CHR$(5)
35 LET CS$=CHR$(147)
40 LET CH$=CHR$(19)
45 POKE BD,6
50 POKE BG,0
55 POKE 650,128
60 PRINT CS$,CC$
100 PRINT TAB(12) "┌─────────────┐"
105 PRINT TAB(12) "│.............│"
110 PRINT TAB(12) "│.○○○.○○○.○○○.│"
115 PRINT TAB(12) "│......○......│"
120 PRINT TAB(12) "│○○○.○...○.○○○│"
125 PRINT TAB(12) "│....○...○....│"
130 PRINT TAB(12) "│.○○...●...○○.│"
135 PRINT TAB(12) "│○●●○.●○●.○●●○│"
140 PRINT TAB(12) "│.○○...●...○○.│"
145 PRINT TAB(12) "│.............│"
150 PRINT TAB(12) "│.○○○○○.○○○○○.│"
155 PRINT TAB(12) "│..○.......○..│"
160 PRINT TAB(12) "│○○○○.○○○.○○○○│"
165 PRINT TAB(12) "│......○......│"
170 PRINT TAB(12) "└─────────────┘"
200 POKE P1,94
210 PRINT "▣FINE DEL GIOCO. PREMERE
   UNO SPAZIO PER CONTINUARE"
215 GET K$
220 IF K$<>"□" THEN 215
225 PRINT "◘◘□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□
   □□□□□□□□□□□"
300 GET A$
305 IF A$="Z" THEN P2=P1-1
310 IF A$="X" THEN P2=P1+1
315 IF A$="P" THEN P2=P1-LL
320 IF A$="L" THEN P2=P1+LL
325 IF PEEK(P2)<>32 AND PEEK(P2)<
   >46 THEN GOTO 350
330 IF PEEK(P2)=46 THEN CO=CO+1
335 POKE P1,32
340 LET P1=P2: POKE P1,94
350 PRINT CH$;
360 PRINT TAB(14);"MONETE:";CO
365 IF CO<103 THEN 300
400 POKE BG,6: CO=0
415 FOR T=1 TO 2000: NEXT T: GOTO 10
```

UN PO' DI COMPETIZIONE

Adesso aggiungere le seguenti linee:

```
205 PRINT TAB(12);"▣MIGLIOR
   TEMPO:□";BT
230 LET TI$="000000"
355 PRINT TAB(2);"TEMPO:";TI$;
400 IF G=0 THEN BT=VAL(TI$): G=1
405 IF VAL(TI$)<BT THEN BT=VAL(TI$)
410 LET CO=0: POKE BG,6
```

MODIFICHE AL LABIRINTO

Desiderando cambiare il labirinto, occorre adattare il contenuto delle PRINT alle linee da 105 a 165. Si può usare qualsiasi simbolo grafico ROM, o anche cambiare le dimensioni del labirinto, ma si rammenti di correggere il valore CO alla linea 365 col nuovo numero di monetine.

METTIAMO ORDINE NELL'ARCHIVIO DEGLI HOBBY

- RICERCA DI UN RECORD
- VARIAZIONE DI UN RECORD
- CANCELLAZIONE DI UN RECORD
- COPIA **SAVE** SU STAMPANTE
- USO DELL'ARCHIVIO

Ricavare informazioni di un archivio gestito da un computer è facile. Si possono anche prevedere modalità di ricerca incrociate o parziali, come per trovare nome e indirizzo di una persona conoscendone solo la città di residenza.

Uno dei maggiori vantaggi dell'archiviazione mediante computer, rispetto a quella tradizionale, è la grande elasticità dei possibili metodi di ricerca, anche conoscendo soltanto una parte dei dati (ad esempio, il nome di battesimo o il numero telefonico di una persona).

Il programma per la gestione di un database, presentanto in questa e nella lezione precedente, è, nella sua semplicità, pur sempre molto flessibile.

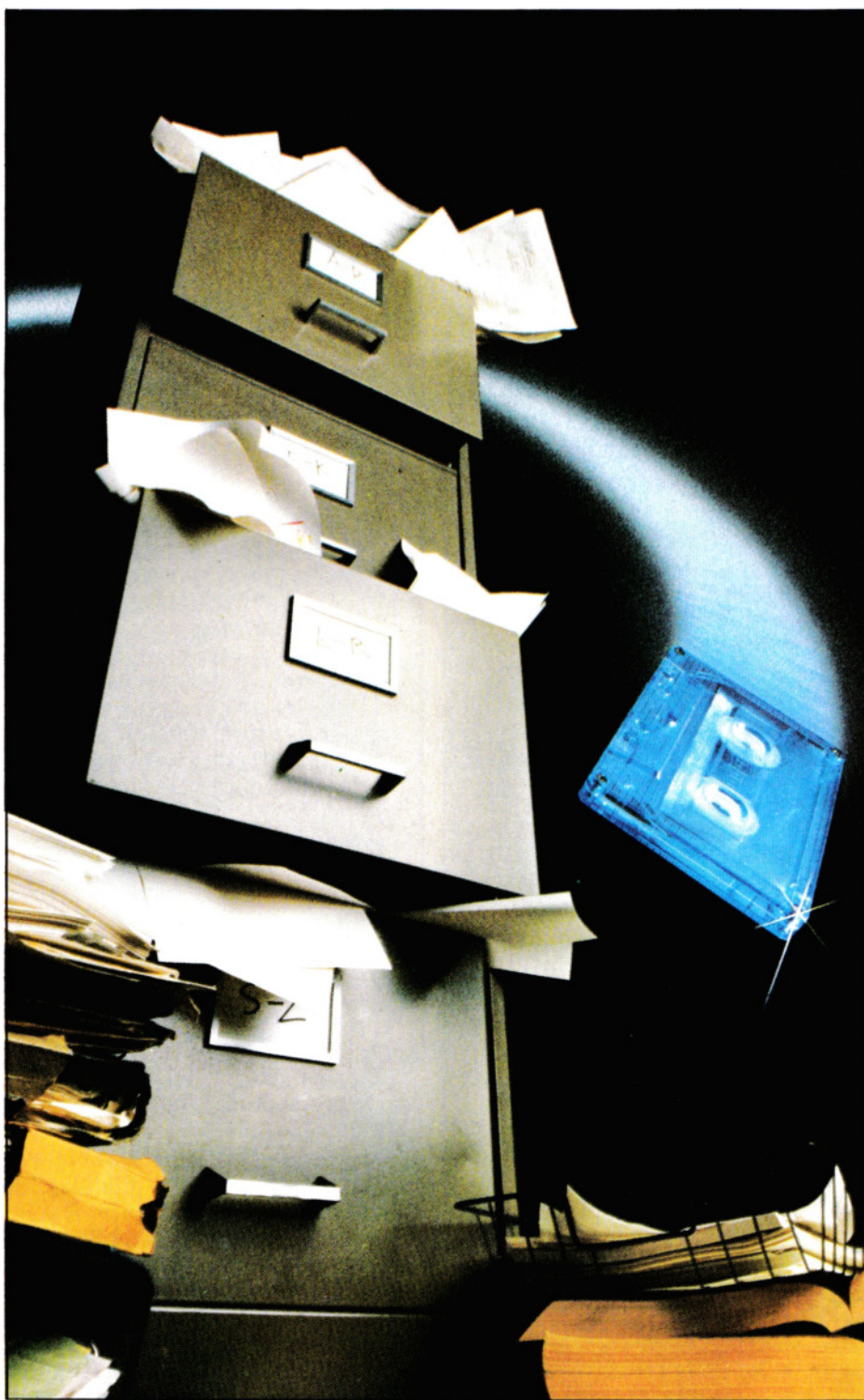
Di seguito, vengono presentate e spiegate le routine di ricerca, di variazione e di annullamento, che completano il programma.

LA RICERCA DI UN RECORD

L'opzione 4, nel menu principale, corrisponde alla ricerca di un record e permette di ricercare una particolare informazione che sia contenuta in uno qualsiasi dei campi.

Premendo il tasto [4], il programma chiede il numero del campo sul quale si intende effettuare la ricerca, dopodiché viene chiesto cosa cercare in quel particolare campo, ad esempio: PIERO, DMU o AMATRICIANA.

L'immissione della *chiave di ricerca* è terminata con [RETURN] o [ENTER].

Ovviamente, la chiave (parola o numero che sia) deve corrispondere esattamente a quanto a suo tempo immesso nel campo: se l'informazione è conservata in caratteri maiuscoli, una ricerca in caratteri minuscoli dà esito negativo. Anche il numero degli spazi usati per separare le parole deve corrispondere: se durante l'immissione si è digitato per errore uno spazio prima o dopo il dato, è probabile che il computer non sia in grado di trovare l'informazione cercata. (Questo rappresenta uno dei casi più ambigui: lo spazio c'è... ma non si vede!)

Conviene adottare un metodo (ad esempio, scrivere sempre i dati in maiuscolo), sia durante l'immissione che durante la ricerca, e poi attenervisi. Ciò evita inutili perdite di tempo.

Qualora il programma non riesca a trovare l'informazione cercata, ci avverte con un messaggio e poi torna al menu principale; invece, nel caso di ricerca positiva, i record trovati vengono presentati in ordine alfabetico.

Durante la fase di ricerca, compaiono due messaggi.

Il primo messaggio è:

A(VANTI) I(NDIETRO) M(ENU)

Anche qui, la scelta avviene secondo il solito sistema: premendo il tasto [A] il programma visualizza il record successivo, mentre premendo il tasto [I] si possono esaminare, ad uno ad uno, quelli precedenti. Il tasto [M] riporta al menu principale.

Se si preme [A] quando si è sull'ultimo record, non accade nulla, mentre una pressione del tasto [I] quando si è sul primo record, provoca l'emissione di un messaggio del tipo 'Record non trovato'.

Esercitarsi in questa operazione di ricerca risulta senz'altro utile a far meglio comprendere le prestazioni e i limiti del programma. Supponiamo che l'archivio sia relativo a un'associazione o a un club e che contenga i seguenti campi: cognome, nome, incarico, via, città, provincia, C.A.P. e numero di telefono. È possibile individuare un nominativo anche ricordandosi solamente il suo incarico o la via in cui abita!

Anche qualora esistessero informazioni identiche, nello stesso campo, ma in più record (ad esempio, più soci si chiamano Mario, e questo nome è usato come chiave di ricerca), non occorre fare altro che scorrere semplicemente i record presentati dal programma, con la certezza di incontrare, prima o poi, quello desiderato. Un simile metodo di ricerca sarebbe davvero complicato, senza un computer.

VARIAZIONE DI UN RECORD

Il secondo messaggio che appare sullo schermo è:

V(ARIAZIONE) C(ANCELLAZIONE) S(TAMPA)

Volendo modificare il contenuto di un record, si preme il tasto [A]. Il computer chiede il numero del campo da variare e, una volta che questo è stato digitato, viene chiesto di introdurre la nuova informazione. Questa deve essere reinserita per intero, anche se si trattasse di variarne una sola lettera, e deve essere terminata con la pressione di un [RETURN] o di un [ENTER].

Il computer colloca automaticamente la variazione nel posto giusto. L'operazione può essere ripetuta, se occorre. Desiderando modificare un altro record, è necessario tornare al menu (tasto [M]), iniziare una nuova ricerca (opzione 3), e, infine, apportare la modifica.

In alternativa, si può scegliere l'opzione 2 (visione record), per posizionarsi sul record e variarlo.

ANNULLAMENTO DI UN RECORD

Anche in questo caso, occorre individuare il record (opzione 2 o 3), prima di poterlo cancellare. In fondo allo schermo, vengono visualizzate le solite sei possibilità A(VANTI), I(NDIETRO), M(ENU), V(ARIAZIONE), C(ANCELLAZIONE), e S(TAMPA). Per cancellare, premere il tasto [C]. Il computer chiede conferma, onde evitare annullamenti accidentali.

La conferma è data con la pressione del tasto [S]. Il computer cancella il record e

presenta quello successivo (quale esso sia dipende dall'opzione in corso: se è la 2, il record è quello alfabeticamente successivo, mentre se è la 3, è quello successivo secondo la chiave di ricerca).

Se si è cancellato l'ultimo record della sequenza, il computer ci avverte che non esistono altri record, riportandoci poi al menu principale.

LA COPIA SU STAMPANTE

L'ultima scelta che rimane da esaminare è la S(TAMPA).

Sul Commodore, sul Dragon, sul Tandy o sugli Acorn, la pressione del tasto S provoca l'emissione del messaggio 'STAMPANTE OK?'.

Se la stampante è pronta all'uso, basta premere il tasto C (continua). In caso contrario, si ha la possibilità di predisporre tutto il necessario.

Se non si possiede una stampante e si prosegue, il programma si blocca. In tal caso, l'unica via d'uscita è premere ESCAPE (sugli Acorn), RUN/STOP (sul Commodore) o RESET (sul Dragon e sul Tandy), terminando il programma.

Per rientrare, si digiti:

```
GOTO 30
```

Sul Commodore, invece, si usi:

```
GOTO 100
```

Sullo Spectrum il problema non sussiste, dato che il computer è in grado di accorgersi da solo se c'è una stampante collegata al sistema.

Digitando qualsiasi altra lettera, diversa dalla C, ci riporta alle sei opzioni standard, sul fondo dello schermo.

USO DELL'ARCHIVIO

Adesso che siamo in possesso di un valido sistema di archiviazione, non resta che scegliere quali dati inserirvi. Naturalmente, lo stesso programma può essere impiegato per gestire più archivi, tutti raccolti in un solo nastro.

Tuttavia, conviene tenere gli archivi separati, uno per cassetta, facilitandone la ricerca.

In tal caso, il programma può essere conservato a parte, su un'apposita cassetta (tranne che nel caso dello Spectrum), inserendo la cassetta dati, prima di scegliere l'opzione 6.

Se questa sembra una soluzione ingombrante, si pensi a quanto spazio occuperebbe l'archivio, se fosse tenuto in maniera tradizionale!

TRASCRIZIONE DEL PROGRAMMA

Di seguito, è riportata la seconda parte del programma: più corta della prima, se ciò può consolare!

Si noterà che, rispetto alla prima parte, alcuni numeri di linea sono ripetuti: non si tratta di un errore, ma semplicemente del fatto che nella versione incompleta le linee in questione devono adesso essere sostituite. Ciò avviene semplicemente digitando le nuove linee:

S

```
4000 FOR N=V TO A: PRINT INVERSE V;AT
    N*2,9;N; INVERSE U; TAB 11;":-□";
    N$(N): NEXT N
4010 PRINT '"RICERCA SU QUALE CAMPO
    (1-□";A;")?"
4020 IF INKEY$="" THEN GOTO 4020
4030 LET Y$=INKEY$: IF CODE Y$<49 OR
    CODE Y$>48+A THEN GOTO 4030
4040 BEEP .1,10: LET Z=VAL Y$: PRINT
    '"Cosa cercate nel campo□";Z;"□?": DIM
    Z$(V,A(Z)): INPUT LINE Z$(V)
4044 CLS: LET K=V
4045 IF A$(K,B(Z)+V TO B(Z+V))=Z$(V)
    THEN GOTO 4050
4046 IF K=R OR A$(K,V)="□" THEN CLS:
    PRINT AT 9,3;"□NESSUN RECORD
    CON□"; Z$(V),'TAB 10;"NEL CAMPO□";
    Z:PAUSE 150:RETURN
4047 LET K=K+V: GOTO 4045
4050 LET D=V: LET PM=V: LET MO=V
4060 IF D>R THEN LET D=PM
4070 IF D=U THEN LET D=PM
4080 IF A$(D,V)="□" THEN LET D=PM
4090 IF A$(D,B(Z)+V TO B(Z+V))<
    >Z$(V) THEN LET D=D+MO: GOTO
    4060
4100 GOSUB 9500
4110 LET PM=D
4120 IF OP=V THEN LET MO=V: LET D
    =D+MO: GOTO 4060
4130 IF OP=2 THEN LET MO=-V: LET D
    =D+MO: GOTO 4060
4140 IF OP=3 THEN RETURN
4150 IF OP=4 THEN GOSUB 8000
4160 IF OP=5 THEN LET DF=U: LET MD
    =2: GOSUB 9000: IF DF=V OR A$(V,V)
    ="□" THEN RETURN
4170 GOTO 4100
8000 INPUT AT U,U;"VARIARE quale campo
    (1-□";(A);") ?";J:IF J>A THEN GOTO
    8000
8010 PRINT FLASH V;AT V+2*J,12;A$
    (D,B(J)+V TO B(J+V)): INPUT AT
    U,U;"Immettere nuovo campo", LINE
    A$(D,B(J)+V TO B(J+V)): PRINT AT V
    +2*J,12;A$(D,B(J)+V TO B(J+V))
8020 IF D=R THEN LET J=-V: GOTO
    8070
8030 IF D=V THEN LET J=V: GOTO 8060
8040 IF A$(D)>A$(D+V) THEN LET J=V
8050 IF A$(D)<A$(D-V) THEN LET
    J=-V
8060 IF A$(D+V,V)="□" AND J=V THEN
    GOTO 8500
8070 IF J=V THEN GOTO 8100
8080 IF A$(D)>=A$(D-V) THEN GOTO
    8500
8090 LET X$=A$(D): LET A$(D)=A$(D
    -V): LET A$(D-V)=X$: LET D=D-V:
    GOTO 8020
8100 IF A$(D)<=A$(D+V) THEN GOTO
    8500
8110 LET X$=A$(D): LET A$(D)=A$(D
    +V): LET A$(D+V)=X$: LET D=D+V:
    GOTO 8020
8500 BEEP .1,10: PRINT AT U,U;"Record
    n.□";D;"□□□": RETURN
```

Perché, dopo la trascrizione, un programma, talvolta piuttosto lungo, non funziona?

Trascrivere lunghi programmi, può essere un compito laborioso, forse scoraggiante. Per alleggerirlo, si può copiare una breve sezione alla volta, per poi confrontarla con l'originale. Alcuni programmi sono già suddivisi in più sezioni o sottoprogrammi. Il metodo più efficace per verificare una singola sezione consiste nel digitare un GOTO, seguito dal numero di linea iniziale del sottoprogramma. Se ora è necessario il valore di qualche variabile usata altrove nel programma, occorre identificarla e attribuirle un valore 'di prova', cosa che può anche rivelarsi difficile. Se non si riescono a individuare sezioni o subroutine nel programma, allora si trascrivano 20 o 30 linee alla volta, con cura. In certi casi, occorre passare a metodi di diagnosi più efficaci che non la semplice rilettura. Ad esempio, individuate le variabili più significative, si possono inserire delle PRINT in vari punti del programma per visualizzarne il contenuto. Questo metodo consente di rilevare se, all'improvviso, una variabile assume un valore insolito o totalmente inesatto.

Le modifiche per l'Electron

Occorre apportare qualche variazione alla versione del Micro BBC, affinché funzioni anche sull'Electron.

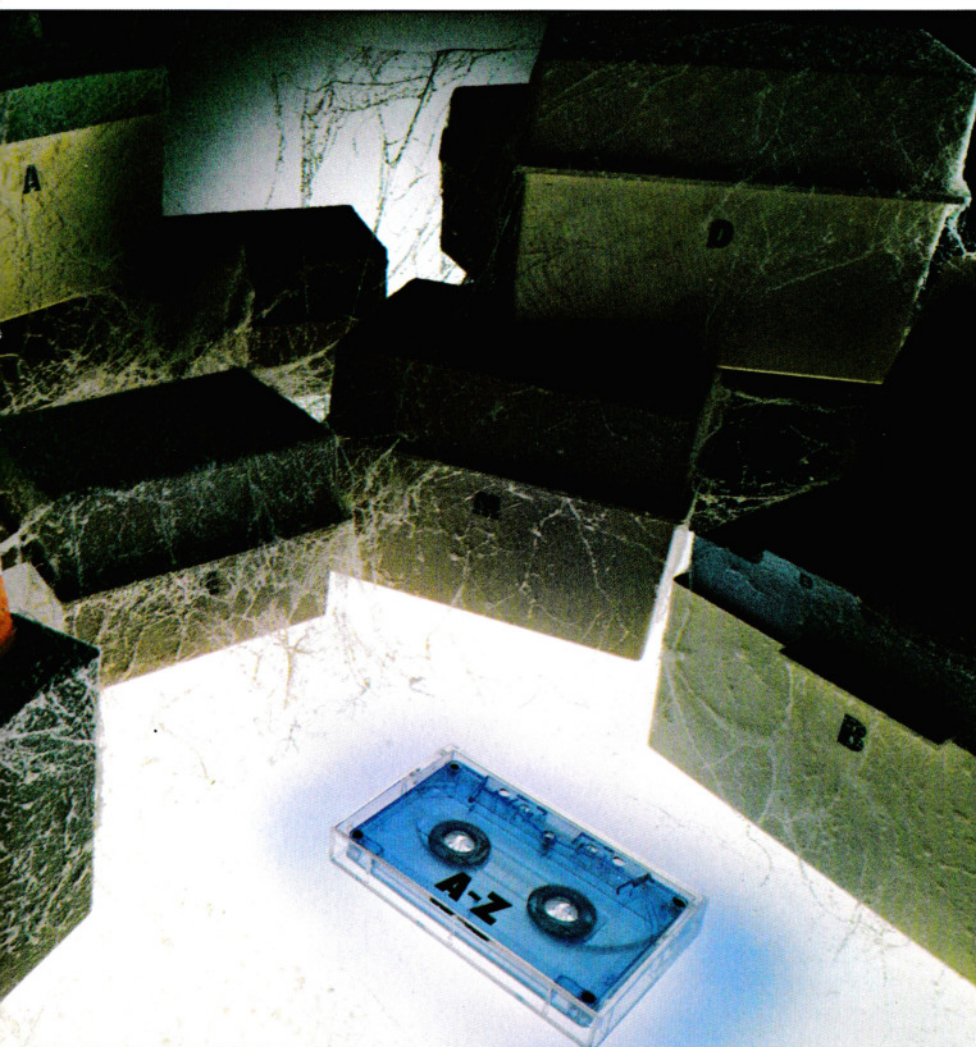
Sostituire le seguenti linee:

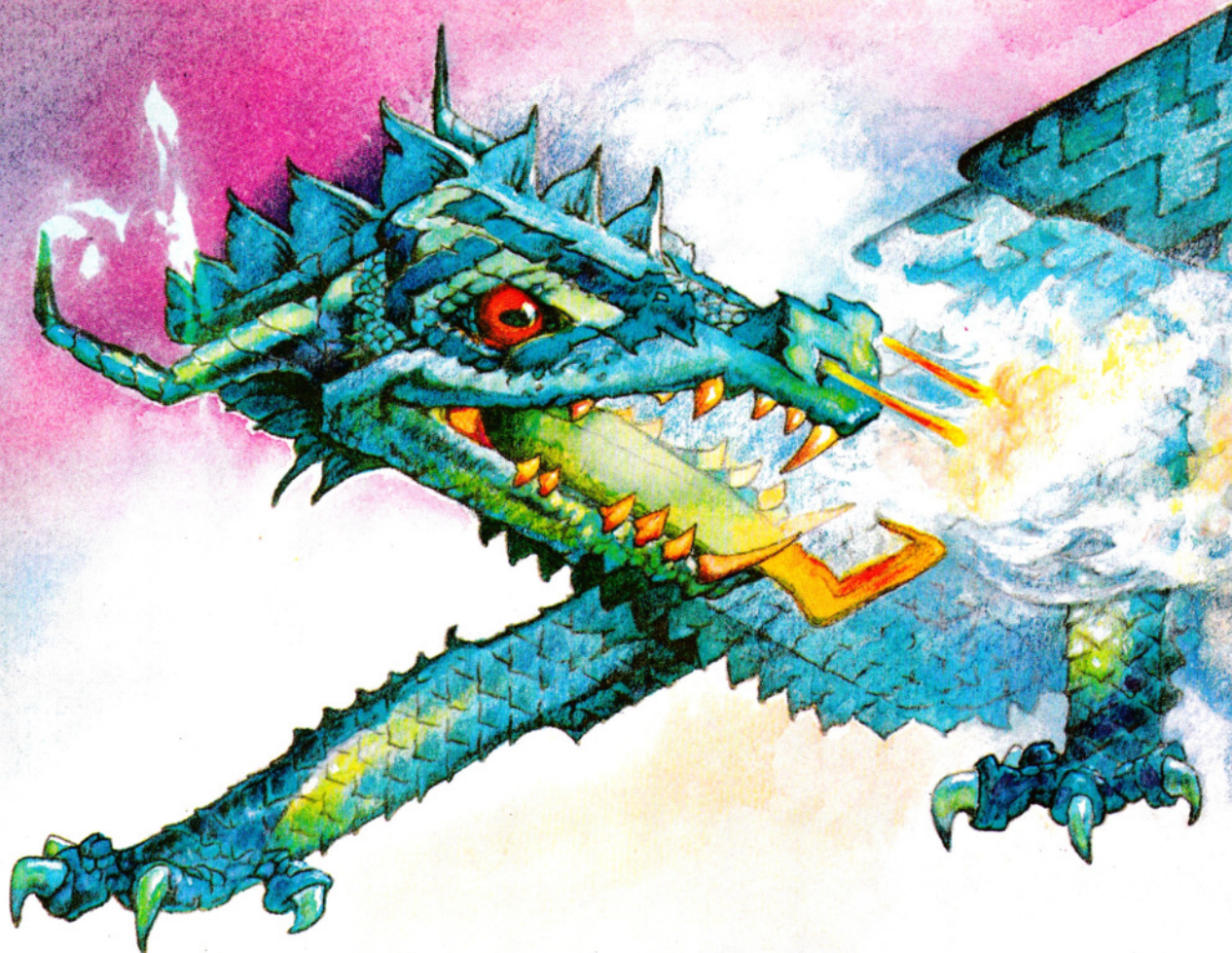
```
5038 Q=-1
5040 PROCFIND
5041 IF C<>0 THEN PROCVDU:
   PROCKEY
5042 UNTIL Q=1 OR C=0
5043 IF Q=1 THEN 5045
5044 CLS:PRINTTAB(12,5)
   "NESSUN□RECORD CON"TAB(20
   -LEN(A$)/2,7)A$TAB(15,9)
   "NEL CAMPO";F:G=INKEY(300)
5045 ENDPROC
5047 DEF PROCFIND
5050 T=0
5060 D%=D%+C
5065 IF D%<1 OR D%>N%-1 THEN C
   =-C:T=T+1:D%=D%+C
5070 IF $B%=$(B%+D%*R%+
   TRL (F)) THEN 5100
5080 IF T<2 THEN 5060
5090 C=0
```

```
9000 PRINT AT 19,U;"CONFERMA DEL
   CANCELLAMENTO (S/N)?": PAUSE U
9010 IF INKEY$="" THEN GOTO 9010
9020 IF INKEY$<>"S" THEN PRINT AT 19,
   U;"□□□□□□□□□□□□□□□□□□
   □□□□□□□□□□
   □□□□□": BEEP .1,10: RETURN
9030 PRINT AT 19,U;;"□
   CANCELLAMENTO IN CORSO□□□□□
   □□□□□□□□□□□□□□□□□□□"
9035 LET DD=D
9040 IF D=R THEN LET DD=DD-V: GOTO
   9060
9050 IF A$(D+V,V)<>"□" THEN LET
   A$(D)=A$(D+V): LET D=D+V: GOTO
   9040
9060 LET A$(D)="": FOR F=V TO 100:
   NEXT F: PRINT AT 19,U;"□□□
   □□□□□□□"
9070 LET D=DD: IF A$(V,V)="□" THEN
   LET D=U: RETURN
9072 IF D=U THEN LET D=V
9075 IF A$(D,V)="□" THEN LET D=D-V
9080 IF MD=V THEN RETURN
9090 LET K=V
9100 IF A$(K,B(Z)+V TO B(Z+V))=Z$
   (V) THEN GOTO 9130
9110 IF K=R OR A$(K,V)="□" THEN LET
   DF=V: GOTO 4046
9120 LET K=K+V: GOTO 9100
9130 LET DD=D: LET PA=V
9140 IF A$(DD,V)="□" OR DD=U THEN
   LET PA=2: LET DD=D: LET MO=MO
   -V
9150 IF A$(DD,B(Z)+V TO B(Z+V))=Z$
   (V) THEN LET D=DD: RETURN
9160 LET DD=DD+MO: GOTO 9140
```

```
3020 PRINT"Numero del campo (1-";A;")?"
3035 R=GET-48:IF R<1 OR R>A□
   THEN 3035
3060 VDU11:PRINT"Immettere modifica□
   □:";: PROCINPUT(A(R)):P.'
3065 IF LEN($B%)<1 THEN 3060
3070 $(B%+D%*R%+TRL(R))=$B%
3080 IF D%=1 THEN3140
3090 X=B%+D%*R%:Y=B%+(D%-1)*R%
3100 IF $X>=$Y□THEN3130
3110 FOR T=1 TOA:$B%=$(X+TRL(T)):$(X
   +TRL(T))=$(Y+TRL(T)):$(Y+TRL(T))
   =$B%:NEXT
3120 D%=D%-1:IF D%=1 THEN 3180
   ELSE 3090
3130 IF D%=N%-1 THEN 3180
3140 X=B%+D%*R%:Y=B%+(D%+1)*R%
3150 IF $X<=$Y□THEN 3180
3160 FOR T=1 TOA:$B%=$(X+TRL(T)):$(X
   +TRL(T))=$(Y+TRL(T)):$(Y+TRL(T))
   =$B%:NEXT
3170 D%=D%+1:GOTO3130
3180 D%=D%-C
4010 VDU11,11:PRINT
   "Conferma cancellamento□?□□";
   STRING$(40,"□")
4020 Q=GET AND &5F:IF Q<>83 THEN
   ENDPROC
4030 IF D%=N%-1 THEN 4080
4050 FOR T=D%+1 TO N%
4060 FOR Q=1 TO A:$(B%+R%*(T-1)+
   TRL(Q))=$(B%+R%*(T)+TRL(Q)):NEXT
4070 NEXT
4080 N%=N%-1:D%=D%-C
5001 IF N%=1 THEN ENDPROC
5005 CLS:FOR N=1 TO A:PRINT';N,N$(N):
   NEXT
5007 D%=1:C=1:Q=0
5010 PRINT'"RIcerca su quale campo (1
   -";A;")?"
5020 F=GET-48:IF F<1 OR F>A□
   THEN 5020
5030 PRINT'"Cosa cercare nel
   campo□";F;"?□";
5035 PROCINPUT(A(F)):A$=$B%:P.'
5037 REPEAT
5040 PROCFIND
5041 IF C<>0 THEN PROCVDU:PROCKEY:Q
   =0
5042 D%=D%+C
5043 UNTIL Q=2:Q=1
5044 CLS:PRINTTAB(12,5)"Nessun record
   con"TAB(20-LEN(A$)/2,7)A$TAB(15,9)
   "nel campo□ ";F:G=INKEY(300
5045 ENDPROC
5047 DEF PROCFIND
```

```
5050 T=0:REPEAT
5070 IF $B%=$(B%+D%*R%+TRL
   (F)) THEN T=2:GOTO 5080
5073 D%=D%+C
5075 IF D%>N%-2 OR D%<2 THEN T=T
   +1:C=-C:Q=Q+1
5080 UNTIL T>1
5090 IF Q=2 THEN C=0
```

```
3000 GOSUB8500
3010 PRINT@417,"QUALE CAMPO VARIARE
   (1 TO";A;")□?";
3020 IN$=INKEY$:IFIN$="" THEN3020
3030 J=VAL(IN$)
3040 IF J<1 ORJ>A THEN3020
3050 PRINT@448,""
3060 PRINT@45+32*J," ":PRINT@417,
   "IMMETTERE NUOVO CAMPO-":
   LINE INPUTA$(D,J)
3070 A$(D,J)=LEFT$(A$(D,J),A(J))
3080 IFD=R THENJ=-1:GOTO3130
3090 IFD=1 THENJ=1:GOTO3120
```

```
3100 IFA$(D,1)>A$(D+1,1) THENJ=1
3110 IFA$(D,1)<A$(D−1,1) THENJ=−1
3120 IFA$(D+1,1)="" AND J=1 THEN
   3180
3130 IFJ=1 THEN3160
3140 IFA$(D,1)>=A$(D−1,1) THEN3180
3150 FORN=1TOA:X$=A$(D,N):A$(D,N)=
   A$(D−1,N):A$(D−1,N)=X$:NEXT:D=D
   −1:GOTO3080
3160 IFA$(D,1)<=A$(D+1,1) THEN3180
3170 FORN=1TOA:X$=A$(D,N):A$(D,N)=
   A$(D+1,N):A$(D+1,N)=X$:NEXT:D=D
   +1:GOTO3080
3180 FORF=1TO300:NEXT:RETURN
4000 G=D
4010 GOSUB8500
4020 PRINT@449,"CONFERMA DEL□□□□
   □□□□□□□CANCELLAMENTO□
   (S/N)?";
4030 IN$=INKEY$:IFIN$=""THEN4030
4040 IFIN$<>"S" THENRETURN
4050 IFG=NR THENG=G−1
4060 CLS7:PRINT@236,
   "STO CANCELLANDO";:SOUND30,1
4070 IFD=NR THEN4090
4080 FORN=1TOA:A$(D,N)=A$(D+1,N):
   NEXT:D=D+1:GOTO4070
4090 FORN=1TOA:A$(D,N)="":NEXT:NR=
   NR−1:D=G
4100 FORF=1TO400:NEXT:RETURN
5000 FORN=1TOA:PRINT@33+32*N,N,N$
   (N):NEXT
5010 PRINT@417,"RICERCA SU QUALE
   CAMPO (1−";A;")□?";
5020 IN$=INKEY$:IFIN$="" THEN5020
5030 IFVAL(IN$)<1 OR VAL(IN$)>A THEN
   5020
5040 SOUND30,1:Z=VAL(IN$):
   PRINT@417,"COSA CERCARE NEL
   CAMPO";Z;"□?"
5050 LINEINPUTZ$
5060 D=1:G=0:CH=1
5070 IFD>NR ANDG=1 THENG=0:CH=
   −1 ELSEIFD>NR THEN5230
5080 IFD<1 ANDG=1 THENG=0:CH=
   1 ELSEIFD<1 THEN5230
5090 IFA$(D,Z)<>Z$ THEND=D+CH:
   GOTO5070
5100 LD=D:G=1:GOSUB8500
5110 PRINT@451,"aVANTI□□□□iNDIETRO
   □□□mENU□□vARIAZIONE□□□□
   cANCELLAMENTO□□□□□□□sTAMPA";
5120 IN$=INKEY$:IFIN$="" THEN5120
5130 IN=INSTR(1,RS$,IN$)
5140 ON IN GOTO 5160,5160,5170,5180,
   5190,5200,5210
5150 GOTO5120
5160 CH=1:D=D+1:GOTO5070
5170 CH=−1:D=D−1:GOTO5070
5180 RETURN
5190 GOSUB3000:GOTO5090
5200 GOSUB4000:GOTO5090
5210 GOSUB10000:GOTO5070
5220 GOTO 5110
5230 CLS2:PRINT@200,
   "NESSUN RECORD CON□";:PRINT@271
   −LEN(Z$)/2,"□";Z$;"□";
5240 PRINT@330,"□NEL CAMPO";Z;
5250 FORG=1TO5:SCREEN0,1: FORF=1TO
   500:NEXT:SCREEN0,0: FORF=1TO500:
   NEXTF,G:RETURN
```

```
3400 ix$=au$(ix):gosub14500
3420 ifaa$="n"thengosub3720:goto3100
3430 u=u−1
3440 ifu=0thenreturn
3450 ru=r(up):ifup=uthen3470
3460 fordn=uptou−1:r(dn)=r(dn+1):next
3470 ifru=uthen3510
3480 fordn=0tou−1:ifr(dn)<>uthen3500
3490 r(dn)=ru:forn=1tonf:t$(ru,n−1)=t$
   (u,n−1):next:dn=u
3500 next
3510 ifup=uthen3980
3520 goto3040
3700 gosub11500:printx2$ro$"□□□□
   □□□□□□□□□□□□□□
   □□□□□□□□□□□□□□
   □□□□□□□□□□□";
3705 print"□□□□NUMERO DEL
   CAMPO□"x3$;
3710 ok$=left$("12345678",nf):gosub10600
3712 gosub3720:goto3800
4000 printcs$x1$gr$"□
   RICERCA SUL CAMPO N.?"cc$"□"cl$;
4010 ok$=left$("12345678",nf):gosub10600
4030 printchr$(ix+48):fx=ix
4040 printx1$gr$"□COSA CERCARE IN
   QUESTO CAMPO?"cc$
4050 y=8:x=0:gosub11500:printro$hd$(ix)
   "□:?";:x=len(hd$(ix))+4
4060 z=lx(ix):gosub13000:ifright$(ix$,1)=
   "□"thenix=left$(ix$,len(ix$)−1)
4070 ifix$=""thengosub10000:goto4050
4080 fx$=ix$:ff=len(fx$)
4100 goto3980
```

CREIAMO UN DRAGO SPUTAFUOCO

La griglia grafica 3 × 3, creata precedentemente, può servire per un'ampia gamma di nuove immagini. Eccone una: uno spaventoso drago che cammina e sputa fuoco!

Un drago è sempre utile per aggiungere un po' di brivido ai giochi d'avventura. In effetti, nessuna storia di maghi e cavalieri sarebbe completa senza un drago.

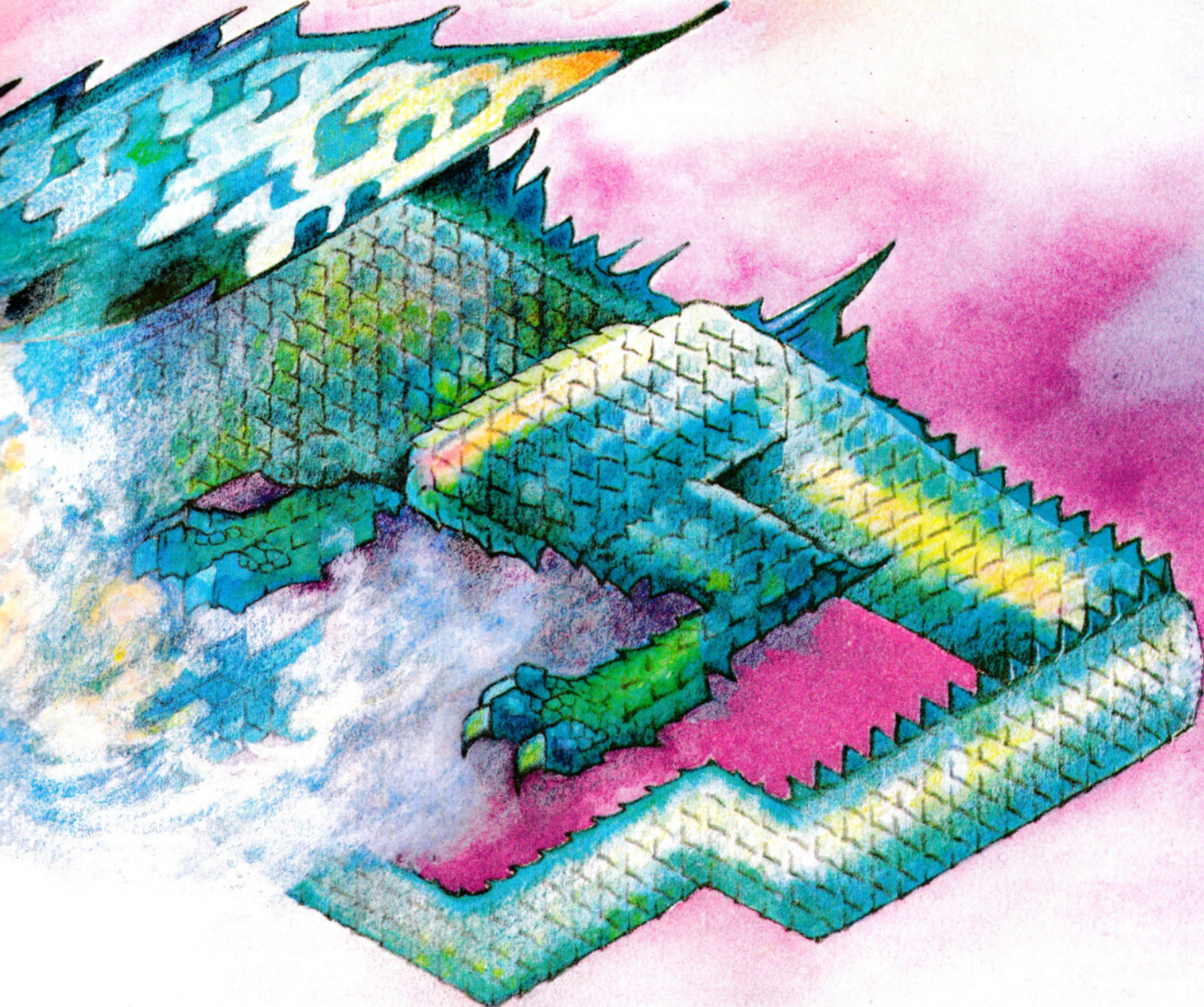
Per niente difficile da creare e da muovere, per un drago si può usare la stessa procedura adottata per il carro armato e per la rana (pagine 8-15). Le routine di movimento sono identiche e il meccanismo per sputar fuoco è ancora più semplice di quello per il lancio dei proiettili nel carro armato.

Prima di creare un drago, va definita una griglia nella memoria del computer. Può essere la stessa usata per creare l'immagine del carro armato e della rana.

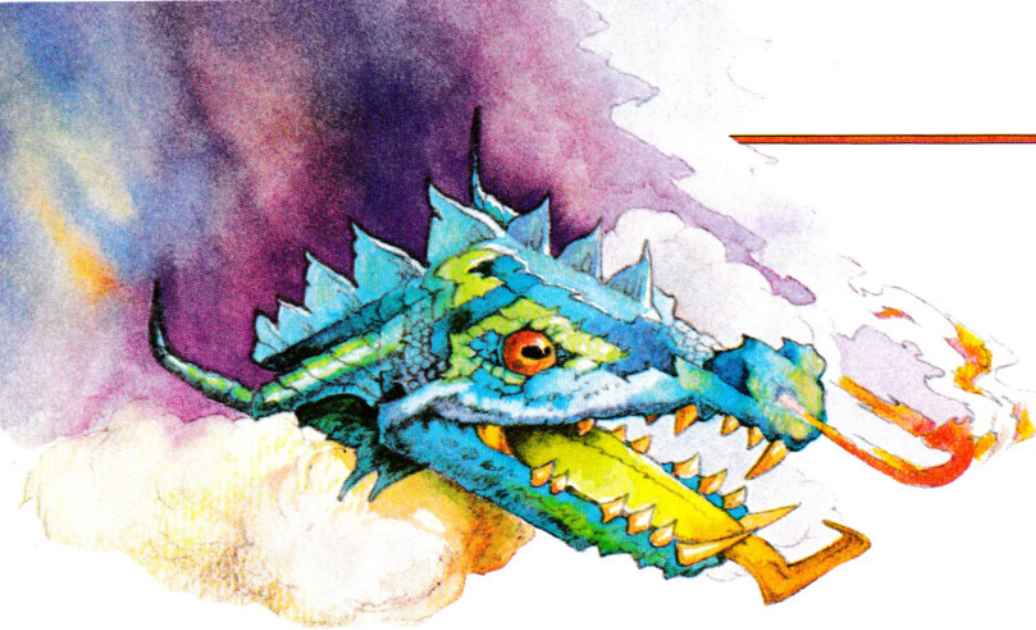
Se si è conservato su nastro il programma per la costruzione della griglia per il carro armato, basterà adesso rileggerlo in memoria e trascrivere il programma per la definizione del drago, tra quelli riportati più sotto. Altrimenti, è necessario, per prima cosa, immettere il programma per la griglia (vedere a pagina 8 per lo Spectrum, a pagina 11 per gli Acorn, e a pagina 13 per il Dragon). Se neanche questa volta si intende salvare questo programma, si possono eliminare le linee che contengono l'istruzione SAVE, ossia la 50 nel programma per lo Spectrum, la 40 in quello per gli Acorn e le linee da 60 a 120 in quello per il Dragon.

Portata a termine la trascrizione, con o senza omissioni, si impartisce un RUN, per depositare il codice macchina in memoria. Fatto questo, si usa il comando NEW (seguito da RETURN o ENTER per ripulire l'area di memoria riservata al BASIC, evitando, al tempo stesso, che il programma del drago venga 'contaminato' da linee 'superstiti'.

A questo punto, si è pronti per trascrivere le linee di programma adatte al proprio computer, prestando particolare attenzione ai valori contenuti in ciascuna frase DATA, prima di premere RETURN o ENTER e passare a quella successiva.

Lanciando il programma, si ottiene sullo schermo l'immagine di un drago. Pos-

■ ADATTARE IN UNA GRIGLIA GRAFICA UN NUOVO PERSONAGGIO
■ MOVIMENTO SULLO SCHERMO
■ COME FAR SPUTAR FUOCO AL DRAGO

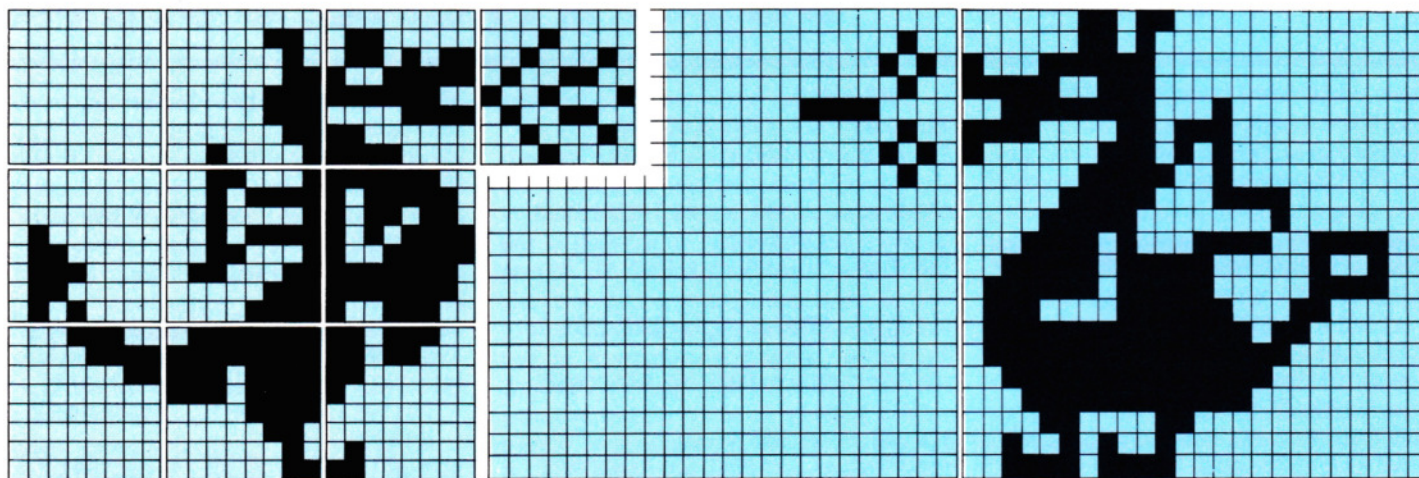
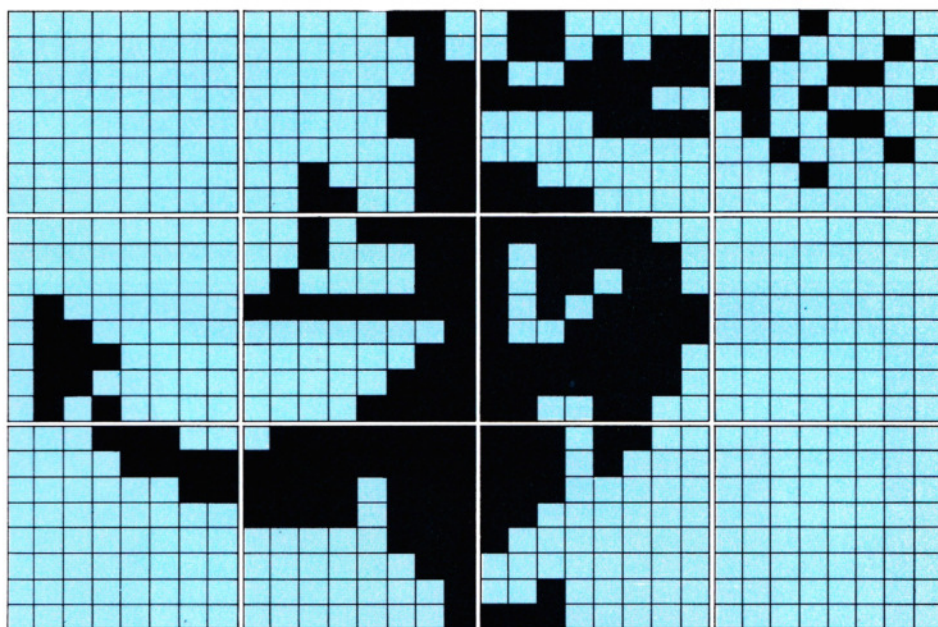
siamo adoperare i tasti [Z], [X], [F] e [L] per muovere il drago, rispettivamente, verso sinistra, verso destra, verso l'alto e verso il basso. Una pressione della barra spaziatrice fa sputar fuoco al drago.

```
10 FOR n=USR "a" TO USR "t"+7: READ
   a:POKE n,a: NEXT n
20 LET print=32400: LET b=32402: IF
   PEEK 23733=255 THEN LET print
   =65200: LET b=65202
90 BORDER 0: PAPER 0: INK 4: CLS: PRINT
   AT 8,15;: RANDOMIZE USR print
100 LET y=8: LET x=15: LET y1=8: LET
   x1=15: LET z=1
110 LET a$=INKEY$: IF a$="" THEN
   GOTO 110
115 IF a$="□" THEN GOTO 200
120 IF a$="z" AND x>1 THEN LET x1=x
   -1: LET z=1
130 IF a$="x" AND x<28 THEN LET x1
   =x+1: LET z=2
140 IF a$="p" AND y>0 THEN LET y1=y
   -1
150 IF a$="l" AND y<19 THEN LET y1
   =y+1
160 PRINT AT y,x;: POKE b,0: RANDOMIZE
   USR print
170 LET x=x1: LET y=y1
180 PRINT AT y,x;: POKE b,z: RANDOMIZE
   USR print
190 GOTO 110
200 IF z=2 THEN GOTO 300
220 PRINT INK 6;AT y,x-1;CHR$ 162
230 PAUSE 1
240 PRINT AT y,x-1;"□"
260 GOTO 110
300 PRINT INK 6;AT y,x+3;CHR$ 163
310 PAUSE 1
320 PRINT AT y,x+3;"□"
330 GOTO 110
1000 DATA 6,214,249,63,240,0,3,15,96,64,
   192,224,224,192,196,204,0,0,0,0,0,0,0
1010 DATA 63,125,107,245,249,127,127,51,
   244,196,194,255,128,192,224,240,0,0,0,2,
   6,14,6,10
1020 DATA 55,23,7,3,1,0,4,7,254,255,239,
   239,224,192,128,128,56,240,192,0,0,0,0,0
1030 DATA 0,0,0,0,0,0,0,0,6,2,3,7,7,3,35,51,
   96,100,159,252,15,0,192,240
```

```
1040 DATA 0,0,0,64,96,112,96,80,47,33,65,
    255,1,3,7,15,252,190,182,175,159,254,
    254,204
1050 DATA 28,15,3,0,0,0,0,0,127,255,247,
    247,7,3,1,1,236,232,224,192,128,0,32,224
1060 DATA 8,68,50,139,50,68,8,0,16,34,76,
    145,76,34,16,0
```

Sul Tandy, occorre sostituire 239 con 251 nella linea 3ØØ, sostituire 247 con 253 nella linea 31Ø e cambiare 223, nelle linee 32Ø e 33Ø, con 247:

```
20 PCLEAR5
30 CLEAR200,32000
40 FORI=32300 TO 32587
50 READ N
60 POKE I,N
70 NEXT
80 DIMA(2),B(2),C(2)
90 PMODE4,1
100 PCLS
110 FORI=1536 TO 1760 STEP32
120 READ N
130 POKEI,N
140 NEXT
150 GET(0,0)-(7,7),A
160 FORI=1536 TO 1760 STEP32
170 READ N
180 POKE I,N
190 NEXT
200 GET(0,0)-(7,7),B
210 PCLS
220 SCREEN1,0
230 T=1
240 DP=3500
250 POKE32700,INT(DP/256)
260 POKE 32701,DP-256*INT(DP/256)
270 POKE32250,T+DT*2
280 EXEC32000
290 LP=DP
300 IFPEEK(338)=239 THENDP=DP-32:
    GOTO360
310 IFPEEK(342)=247 THENDP=DP+32:
    GOTO360
320 IFPEEK(340)=223 THENDP=DP-1:T
    =T+1:DT=0:GOTO360
330 IFPEEK(338)=223 THENDP=DP+1:T
    =T+1:DT=0:GOTO360
340 IFINKEY$="□" AND T=2 GOSUB410
350 GOTO300
360 IDP<1536 OR DP>6941 THENDP=LP
370 IFT>2 THENT=1
380 POKE32250,0
390 EXEC3200
400 GOTO250
410 IF DT=  GOTO470
420 YP=INT((DP-1536)/32)
430 XP=8*(DP-1533-YP*32)
440 IFXP>255 THEN530
450 PUT(XP,YP)-(XP+7,YP+7),A
460 GOTO510
470 YP=INT((DP-1536)/32)
480 XP=8*(DP-1537-YP*32)
490 IFXP<0 THEN530
500 PUT(XP,YP)-(XP+7,YP+7),B
510 FORI=1TO200:NEXT
520 PUT(XP,YP)-(XP+7,YP+7),C
530 RETURN
540 DATA 0,0,0,0,0,0,0,0,6,2,3,7,7,3,35,51,96,
    107,159,252,15,0,192,240
550 DATA 0,0,0,0,64,96,112,96,47,33,65,255,
    3,7,15,31,248,188,182,175,159,255,254,
    254.
560 DATA 80,28,15,7,1,0,0,0,63,255,255,255,
    255,15,2,3,252,248,252,252,248,48,82,222
570 DATA 0,0,0,0,0,0,0,0,0,6,2,3,7,7,3,33,0,96,
    107,159,252,15,192,240
580 DATA 0,0,0,64,96,112,96,80,49,47,33,47,
    49,99,7,15,252,190,182,175,159,254,254,
    156
```

1. Sul Dragon (e Tandy), l'immagine è ottenuta con un UDG standard

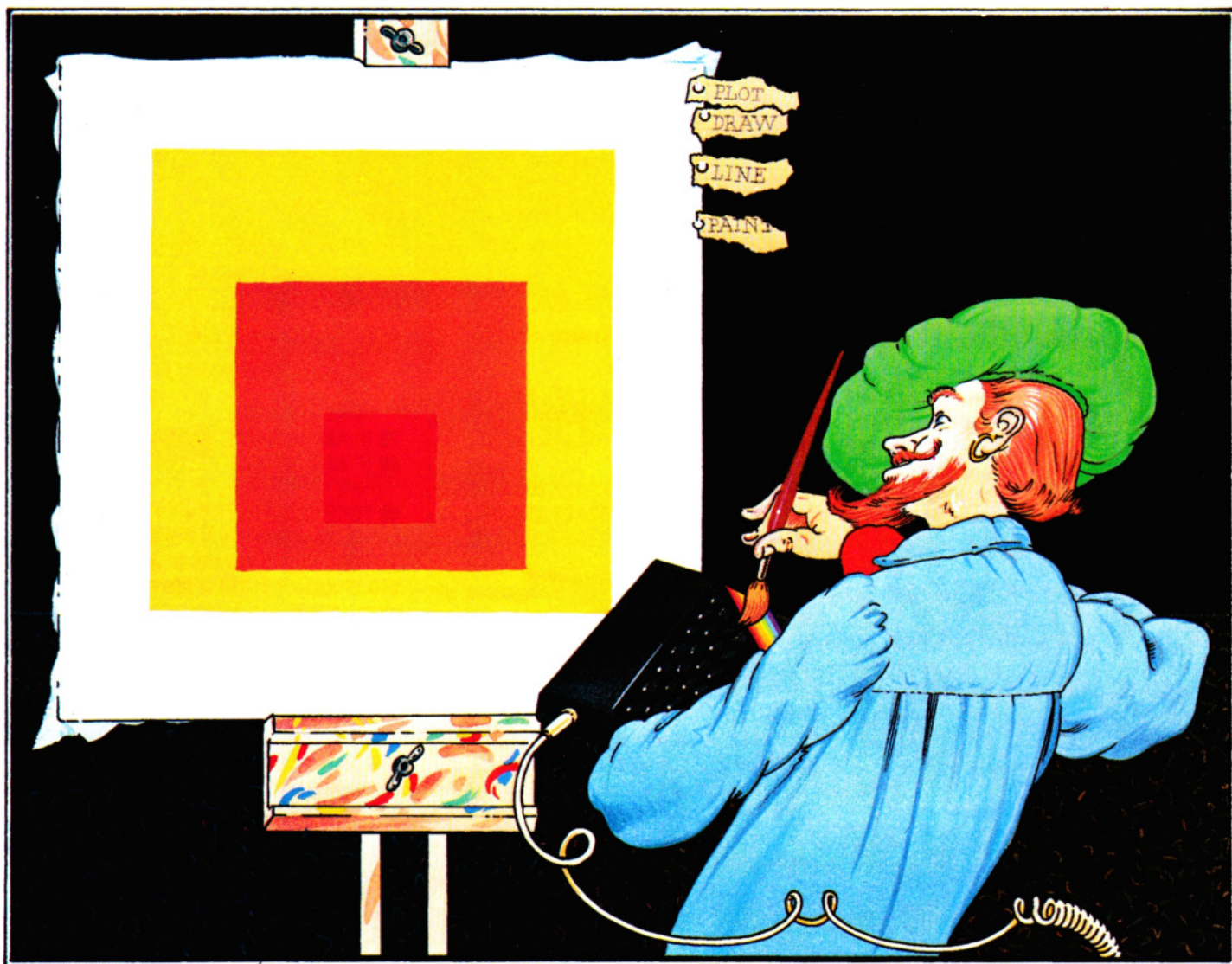
2. La versione Commodore impiega gli sprite a disposizione

```
590 DATA 28,15,3,0,0,0,0,0,127,255,239,239,
    15,6,2,3,220,216,192,128,0,0,64,192
600 DATA 6,214,249,63,240,0,3,15,96,64,192,
    224,224,192,196,204,0,0,0,0,0,0,0,0,
610 DATA 31,61,109,117,249,255,127,127,
    244,196,194,255,192,224,240,248,0,0,0,
    0,2,6,14,6
620 DATA 63,31,63,63,31,12,74,123,252,255,
    255,255,255,240,64,192,10,56,240,224,
    128,0,0,0
630 DATA 0,6,214,249,63,240,3,15,0,96,64,
    192,224,224,192,132,0,0,0,0,0,0,0,0
640 DATA 63,125,109,245,249,127,127,57,
    140,244,132,244,140,198,224,240,0,0,0,2,
    6,14,6,10
650 DATA 59,27,3,1,0,0,2,3,254,255,247,247,
    240,96,64,192,56,240,192,0,0,0,0,0
660 DATA 0,16,34,76,145,76,34,16
670 DATA 0,8,68,50,137,50,68,8
```

3. Ecco il drago costruito con gli UDG dello Spectrum e degli Acorn

```
10 MODE1
20 VDU23;8202;0;0;0;
30 FOR T=224 TO 243:VDU23,T
40 FOR P=1 TO 8
50 READ A
60 VDUA:NEXT
70 NEXT:VDU 19,3,2,0,0,0,:CALL&D00
80 X=0:Y=0:X2=0:Y2=0:Z=1
90 A$=GET$
100 IF A$="Z" AND X>0 THEN X2=
    X-1: Z=1
110 IF A$="X" AND X<37 THEN X2=
    X+1: Z=2
120 IFA$="L" AND Y<29 THEN Y2=
    Y+1
130 IF A$="P" AND Y>0 THEN Y2=
    Y-1
140 IF A$="□" THEN 200
150 VDU31,X,Y:X%=0:CALL &D00
160 X=X2:Y=Y2
170 VDU31,X,Y
180 X%=Z:CALL &D00
190 GOTO 90
200 IF Z=2 THEN 240
210 VDU31,X-1,Y,242,8
220 A$=INKEY$(5):VDU32
230 GOTO 90
240 VDU31,X+3,Y,243,8
250 A$=INKEY$(5):VDU32
260 GOTO 90
270 DATA 6,214,249,63,240,0,3,15,96,64,192,
    224,224,192,196,204,0,0,0,0,0,0,0,0
280 DATA 63,125,109,245,249,127,127,51,
    244,196,194,255,128,192,224,240,0,0,0,2,
    6,14,6,10
290 DATA 55,23,7,3,1,0,4,7,254,255,239,239,
    224,192,128,128,56,240,192,0,0,0,0,0
300 DATA 0,0,0,0,0,0,0,0,6,2,3,7,7,3,35,51,96,
    107,159,252,15,0,192,240
310 DATA 0,0,0,64,96,112,96,80,47,33,65,
    255,1,3,7,15,252,190,182,175,159,254,
    254,204
320 DATA 28,15,3,0,0,0,0,0,127,255,247,247,
    7,3,1,1,236,232,224,192,128,0,32,224
330 DATA 8,68,50,139,50,68,8,0,16,34,76,
    145,76,34,16,0
```

L'uso della grafica definita dall'utente (UDG), sul Commodore 64, differisce leggermente da quello degli sprite (o griglie grafiche) già esistenti nella memoria dell'apparecchio. Di conseguenza, non è necessario lanciare un programma per la costruzione preliminare di una griglia, prima di trascrivere il programma del drago, riportato sotto; basta immetterlo e dargli un RUN:

```
10 SC=53248:X=150:Y=157:ZZ=+45:
    POKE 650,255:PRINT "♥":POKE53281,0:
    POKE SC+39,5
20 FOR I=16000 TO 16254:READ A:POKE I,
    A:NEXT I
30 POKE 2040,250:POKE SC+29,255:
    POKE SC+23,255:POKE 2041,252
40 GET A$:POKE SC+21,253:IF A$=
    "□" THEN POKE SC+2,X+ZZ:POKE SC
    +3,Y:POKESC+21,255
50 IF A$="Z" AND X>60 THEN X=X-2:
    POKE 2040,251:POKE 2041,253:ZZ=-
    45
60 IF A$="X" AND X<200 THEN X=X+
    2:POKE 2040,250:POKE 2041,252:ZZ=
    +45
70 IF A$="P" AND Y>70 THEN Y=Y-2
80 IF A$="L" AND Y<200 THEN Y=Y
    +2
90 POKE SC,X:POKE SC+1,Y
100 IF A$="□" THEN POKE SC+2,X+ZZ:
    POKE SC+3,Y:POKE SC+21,255
110 IF A$="□" THEN POKESC+40,RND
    (1)*8
120 GOTO 40
130 DATA0,6,192,0,2,192,0,3,241,0,3,159,0,
    35,252,0,59,15,0,43,129,0,39,192
140 DATA1,227,224,1,1,240,60,241,120,36,
    31,124,60,31,126,6,63,142,3,127,254
150 DATA1,255,254,0,255,252,0,127,248,0,
    12,48,0,4,164,0,7,188,0
160 DATA3,96,0,3,64,0,143,192,0,249,192,0,
    63,196,0,240,220,0,129,212,0,3,228,0
170 DATA7,199,128,15,128,128,30,143,60,62,
    248,36,126,248,60,112,252,96,127
180 DATA254,192,127,255,128,63,255,0,31,
    254,0,12,48,0,37,32,0,61,224,0,0
190 DATA0,0,0,32,0,0,80,0,0,32,0,0,15,0,0,32,
    0,0,80,0,0,32,0,0,0,0,0,0,0,0,0
200 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0
210 DATA0,0,0,0,0,4,0,0,10,0,0,4,0,0,240,0,0,
    4,0,0,10,0,0,4,0,0,0,0,0,0,0
220 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

I COMANDI PLOT, DRAW LINE E PAINT

Un paio di semplici comandi e un po' di fantasia: ecco tutto ciò che serve per dipingere sullo schermo del computer. Vediamo da dove si comincia

Collegando un televisore o un monitor al computer, possiamo disporre di una *tela* sulla quale *disegnare e dipingere,* ottenendo, con un po' di pazienza e attenzione, anche immagini di grande effetto.

La grafica computerizzata ha fatto passi da gigante in questi ultimi anni, trovando applicazione in tutti i campi. Grafici e diagrammi sono l'esempio più noto, ma anche molte immagini e scritte pubblicitarie, nonché alcuni film, sono prodotti usando il computer. Certo, gli apparecchi utilizzati in tali applicazioni sono di dimensioni molto maggiori dei normali home computer. Tuttavia, anche questi ultimi possono produrre cose strabilianti...

La maggior parte degli attuali home computer sono dotati di comandi del tipo PLOT, DRAW, PAINT, INK e simili. Questi permettono di tracciare linee, di colorarle o riempire di colore intere zone dello schermo. Nel funzionamento grafico, lo schermo viene suddiviso in minuscole unità, chiamate 'pixel', che possono venir illuminate, colorate o spente. In alcuni computer si può scegliere tra un'alta risoluzione, in cui i pixel sono piccoli e le linee molto sottili, o una bassa risoluzione, in cui vale l'opposto. In genere, più alta è la risoluzione, minore è la scelta dei colori.

Dunque, il tracciamento di una linea si ottiene facendo illuminare una serie di pixel. Dopo qualche esperimento con semplici disegni, ognuno scopre in sé l'artista.

Lo ZX81 ha una grafica piuttosto limitata, perciò i programmi presentati per questo apparecchio esplorano l'unico comando disponibile, il PLOT. Nemmeno i Commodore hanno comandi BASIC per la gestione della grafica ad alta risoluzione, tuttavia esistono eccellenti programmi, su cartuccia, che aggiungono tali funzioni.

- USO DEI COMANDI DEL **BASIC**
- COME DISEGNARE QUADRATI, TRIANGOLI E CERCHI
- COLORI E SFUMATURE
- IL COMANDO **FLASH** DELLO SPECTRUM RAVVIVA I PROGRAMMI
- IL SIMON'S BASIC E LA GRAFICA DEL COMMODORE 64
- I VARI MODI GRAFICI DEGLI ACORN

Lo Spectrum ha una risoluzione di 256 pixel in larghezza e di 176 pixel in altezza. Ognuno di questi è numerato secondo un criterio che, a prima vista, può sembrare strano. Chi ha confidenza con le istruzioni PRINT AT, sa che:

```
PRINT AT 10, 15....
```

serve per visualizzare quanto segue l'istruzione nella 10ma riga dall'alto e nella 15ma colonna da sinistra.

Al contrario, nel comando PLOT, riferito a singoli pixel, il primo numero indica la distanza *da sinistra* e il secondo la distanza *dal basso* dello schermo:

```
PLOT 10, 15
```

serve, infatti, per illuminare un punto 10 pixel da sinistra e 15 righe dal basso (si provi il comando: si vedrà un puntino nell'angolo in basso a sinistra).

Le posizioni orizzontali sono numerate da 0 a 255 e quelle verticali da 0 a 175. A titolo di esempio, è utile provare, una alla volta, le seguenti linee:

```
PLOT 0, 0
PLOT 0, 175
PLOT 255, 0
PLOT 255, 175
```

I quattro punti appena illuminati delimitano l'area utilizzabile per un disegno. La zona esterna costituisce la 'cornice' (BORDER - vedere più sotto). Il comando PLOT si può usare con qualsiasi colore.

TRACCIAR LINEE

Il comando DRAW, sullo Spectrum, disegna linee, purché se ne definiscano il punto di partenza, la lunghezza e la direzione.

Il comando PLOT può essere usato per definire il punto iniziale. Si provino le seguenti linee, ma senza ripulire lo schermo tra l'una e l'altra:

```
PLOT 100, 75
DRAW 50, 0
```

Se non viene indicata una nuova posizione, con PLOT, il computer prosegue dalla posizione in cui si trova, ossia, al termine della linea appena tracciata. Per dimostrare ciò, si usino i seguenti comandi, uno alla volta:

```
DRAW 0, 50
DRAW -50, 0
DRAW 0, -0
```

Come si noterà, si adoperano numeri positivi per disegnare verso l'alto o verso destra e numeri negativi per disegnare verso il basso o verso sinistra. Per le linee oblique, la regola è identica: prima si stabilisce di quanti punti essa deve muoversi a destra (o a sinistra), poi di quanti punti verso l'alto (o il basso). Si aggiunga, per esempio, questa linea alle precedenti:

```
DRAW 50, 50
```

ADESSO, UN PROGRAMMA

Ecco qualcosa di più interessante, ottenuto con la DRAW. Per vederne l'effetto, si trascriva e si esegua questo programma:

```
10 INK 2
20 PLOT 0, 175
30 LET t=255: LET r=-175: LET b=
   -255: LET l=169
40DRAW t, 0: DRAW 0, r: DRAW b, 0: DRAW
   0, l
```

Se non si sono commessi errori (aver confuso la elle minuscola (l) con un 1, ad esempio), si ottiene una scatola senza un angolo. Aggiungiamo ora queste linee:

```
50 LET t=t-6
60 LET r=r+12: LET b=b+12: LET l=l
-12
70 DRAW t, 0: DRAW 0, r: DRAW b, 0:
   DRAW 0, l
```

Questo, provare per credere, disegna un'altra scatola interna alla prima. Aggiungiamo ora il resto del programma:

```
80 LET t=t-6
90 GOTO 50
```

Ciò che accade in mezzo allo schermo può forse sorprendere. La ragione è semplice: ad ogni iterazione del ciclo (linee 50 e 90) viene sottratto 12 a ciascuno dei numeri positivi. La variabile t (da top, posizione dall'alto) diminuisce da 255 a 243, a 231, ecc. Se il numero diventa negativo, la linea sotto il suo controllo cambia direzione. Un processo analogo accade per le variabili negative quali la b (da bottom, posizione dal basso).

Per meglio apprezzare questo procedimento, possiamo inserire una breve pausa:

```
85 PAUSE 100
```

lanciando nuovamente il programma.

COLORI E SFUMATURE

Lo Spectrum non dispone di un comando COLOUR col quale ottenere facilmente sfumature e colori sullo sfondo.

È possibile usare una sequenza di linee parallele, ma, poiché per ciascuna linea occorre una PLOT, il programma diventa molto laborioso e ingombrante.

Un sistema per aggirare l'ostacolo è quello di adoperare un ciclo FOR ... NEXT. Nel seguente esempio viene tracciata la parete di destra della casa in figura 2:

```
20 FOR h=56 TO 100 STEP 3
30 PLOT 160,h
40 DRAW 23,12
50 NEXT h
60 PLOT 183, 110: DRAW 0, -97: DRAW
   -23, -13: DRAW 0,100
```

Il funzionamento è semplice: a ogni iterazione del ciclo, il programma disegna una linea lunga 23 pixel, *scalando* da sinistra a destra di 12 pixel per volta. A ogni giro, inizia 160 pixel da sinistra.

Ma, a ogni nuovo 'giro', la linea 20 somma 3 alla variabile h (da height), che rappresenta l'altezza per la posizione di PLOT, cosicché ogni linea è di 3 pixel più alta della precedente.

Con le seguenti linee, si ottiene gran parte del tetto:

```
70 LET r=100
80 FOR p=140 TO 150
90 PLOT r,p
100 DRAW 69, -44
110 LET r=r+2
120 NEXT p
```

Qui troviamo due variabili: p e r. Gli spostamenti di PLOT, infatti, devono avvenire non solo verso l'alto, ma anche verso destra. Tra parentesi, si noti l'uso di valori negativi nella linea 100, per dare al tetto un'inclinazione verso il basso. Queste poche righe completano i contorni della casa:

```
130 PLOT 100,140
140 DRAW −60, −40
150 PLOT 46, 103
160 DRAW 0, −96
170 DRAW 113, −7
```

Adesso possiamo, a nostro piacimento, aggiungere una porta, delle finestre e un comignolo, usando PLOT e DRAW. Ecco alcuni suggerimenti utili:

- Il triangolo sfalsato, sotto il tetto, richiede *tre* variabili: una controlla l'altezza di partenza, una la lunghezza e l'ultima il punto di partenza, a destra, di ciascuna linea.
- Le finestre del piano superiore sono *scavate* nel triangolo. Per far ciò è necessario visualizzare degli spazi, in opportune posizioni, usando PRINT AT. (È utile il metodo descritto in Giochi al computer 1). Infine, si ripassano i contorni, con i soliti PLOT e DRAW.

DISEGNARE CERCHI

Sullo Spectrum, c'è un'apposita istruzione per il disegno dei cerchi: CIRCLE.

Tutto ciò che occorre ricordare è che i primi due valori indicati dopo CIRCLE individuano il centro del cerchio (che non viene visualizzato), mentre il terzo stabiliscie la lunghezza, in pixel, del raggio. Ecco un esempio:

```
CIRCLE 127, 87, 50
```

Questo comando disegna un cerchio di 100 pixel diametro (50 di raggio) al centro dello schermo.

Nel programma per disegnare la casa, si potrebbe usare un comando simile per *aprire* una finestra circolare sulla porta d'ingresso.

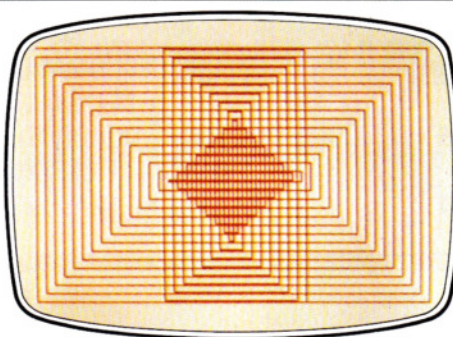
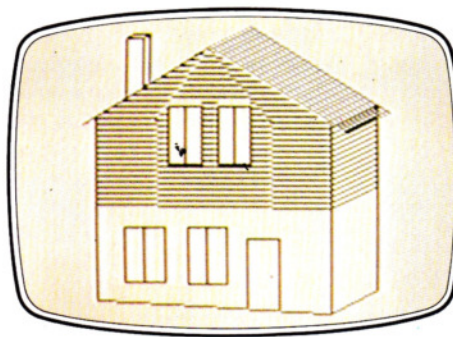
Se CIRCLE viene eseguito da un comando DRAW, la linea che si ottiene inizia dal punto nel quale si è concluso il tracciamento del cerchio.

I COLORI DELLO SPECTRUM

Lo Spectrum ha una gamma di otto colori, il cui codice (da 1 a 7) è riportato sui tasti, tramite i quali si possono richiamare.

Il comando BORDER controlla l'area esterna a quella consentita per il disegno, dove non si possono usare né DRAW né PRINT. Gli altri comandi per i colori hanno varie applicazioni. Eccone una:

```
10 BORDER 2: PAPER 2: INK 7
```

Usando questa linea in un programma, tutte le PRINT che seguono provocano una visualizzazione in bianco, su sfondo rosso, finché non si alterino nuovamente i colori (ciò comprende anche i listati, che talvolta possono risultare illeggibili).

1. Uno schema ottenuto su Spectrum

2. Si può anche dare una prospettiva

Ma se usiamo, invece, una linea del tipo:

```
10 PRINT PAPER 2; INK 7; AT 15,10; "*"
```

allora solo la piccola porzione di schermo dove appare l'asterisco si colora di rosso.

Si noti che, nel primo esempio, i comandi di colore sono seguiti da un 'due punti', nel secondo esempio da un punto e virgola. Queste istruzioni sui colori possono venir incorporate nelle linee che contengono PLOT, DRAW e CIRCLE, Si provino le seguenti linee, ripulendo lo schermo CLS, tra l'una e l'altra:

```
PLOT PAPER 1; INK 7; 125, 87
```

(Si riesce a vedere il punto?)

```
PLOT 125, 87: DRAW INK 2; 50, 50
PLOT 125, 87: DRAW PAPER 1; INK 7; 50,50
CIRCLE INK 4; 127, 87, 50
```

... e poi:

```
CIRCLE PAPER 2; INK 6; 125, 87, 50: DRAW
    PAPER 3; INK 7; 75,0
```

L'ultima combinazione può esser usata per produrre effetti spettacolari, (per quanto, di scarsa utilità pratica).

Altrettanto spettacolare, e molto adoperato nei programmi di gioco, è l'ultimo dei comandi Spectrum trattato in questa lezione: FLASH. La sua funzione è quella di alternare rapidamente i colori definiti da INK e PAPER. La funzione FLASH viene attivata facendola seguire da un 1, e disattivata con uno 0. Ecco un esempio del suo impiego:

```
10 PAUSE 0
20 IF INKEY$ = "" THEN GOTO 20
30 FOR n = 10 TO 23: BEEP .015,n: NEXT n
40 BORDER 2: PAPER 2: INK 6: FLASH 1
50 PRINT "NON TOCCARE!!!"
60 PRINT "QUESTO COMPUTER È
    PERICOLOSO!!!"
70 PRINT
80 GOTO 30
```

Trascritto il programma, lo si lanci, poi, si prema un tasto qualsiasi allontanandosi dal computer!

5

Lo ZX81 usa un comando PLOT smile a quello dello Spectrum. PLOT è seguito da due numeri, che controllano la posizione del pixel. Ma c'è una differenza: i pixel dello ZX81 sono 16 volte più grandi di quelli dello Spectrum, cosicché vi sono soltanto quattro pixel in ogni spazio-carattere, contro i 64 dello Spectrum.

Di conseguenza, non si possono tracciare disegni altrettanto ben definiti (in compenso, almeno i pixel si vedono bene!).

Il seguente programma traccia un'immagine sullo schermo con PLOT. Poiché contiene un ciclo senza fine, che fa ripartire l'esecuzione dalla linea 70, per uscirne occorre premere il tasto BREAK:

```
10 LET X = INT (RND*32)
20 LET Y = INT (RND*24)
30 PLOT X,Y
40 PLOT 63 − X,Y
50 PLOT X,43 − Y
60 PLOT 63 − X, 43 − Y
70 GOTO 10
```

Lo ZX81 non possiede il comando DRAW, ma con un ripetuto uso di PLOT, si possono tracciare delle linee. Nel seguente esempio, mediante due cicli FOR ... NEXT, vengono disegnati dei quadrati:

```
10 FOR N = 0 TO 20 STEP 2
20 FOR M = N TO 43 − N
30 PLOT M,N
40 PLOT M, 43 − N
50 PLOT N,M
60 PLOT 43 − N,M
70 NEXT M
80 NEXT N
```

Similmente, si può simulare il comando CIRCLE con questa breve subroutine:

```
10 FOR N = 0 TO 2* PI STEP .1
20 PLOT 32 + 20*SIN N, 22 + 20*COS N
30 NEXT N
```

In questo sottoprogramma, vengono usate le funzioni trigonometriche SIN e COS, per calcolare le coordinate dei punti del cerchio. Una successiva lezione tratta queste funzioni. Per il momento, si possono variare le dimensioni del cerchio, modificando i due 20 alla linea 20. Il massimo valore è 22, altrimenti l'immagine esce dallo schermo. Oppure, cambiare la posizione del cerchio. Nell'esempio, il centro è situato a 32, 22, come si vede alla linea 20. Ricordarsi di rimpicciolire il cerchio se ci si avvicina ai margini dello schermo.

Benché il Commodore 64 abbia varie e, potenzialmente, molto sofisticate funzioni per la grafica, non è previsto un modo per usare le più spettacolari direttamente dal BASIC. In pratica, si può soltanto accedere ai simboli grafici a bassa risoluzione e ai caratteri definiti nelle ROM.

Tutto risulta più facile, se si acquista una cartuccia aggiuntiva per il Commodore 64, chiamata il 'Simon's BASIC', che contiene un programma atto a facilitare l'impiego di tutti gli effetti grafici e sonori disponibili sul Commodore.

Per la grafica, vengono aggiunte ben diciotto parole chiave (o comandi); tutte, tranne COLOUR, devono essere però usate all'interno dei programmi: non è consentito usarle in modo diretto, come accade, per esempio, con i normali LIST o PRINT.

Inoltre, per eseguire i programmi che ricorrono a questi comandi speciali, occorre inserire nel computer la cartuccia.

PRIMI PASSI COL SIMON'S BASIC

Accendendo il computer, con la cartuccia inserita, lo schermo appare bianco e con un bordo blu.

Le scritte compaiono in maiuscolo e in nero, come il messaggio che informa su quanta memoria è riservata all'estensione del BASIC.

L'unico comando che si può provare in modo diretto è:

```
COLOUR 5,5
```

I valori che seguono COLOUR definiscono il colore dello sfondo e quello del bordo dello schermo. Il valore 5, in ambedue i casi, corrisponde al colore verde (si consulti il manuale per gli altri colori). Le scritte sono sempre in nero. HIRES è l'istruzione per attivare l'alta risoluzione e i valori che ad essa seguono definiscono il colore per i disegni e quello per lo sfondo (per l'elenco completo dei colori, consultare il manuale). Si noti che *deve* essere usato uno spazio separatore nell'istruzione: Questa linea predispone il computer al tracciamento, in alta risoluzione, di disegni in bianco su nero. Eseguendo, lo schermo diventa immediatamente nero.

Il primo dei comandi per disegnare è REC, la cui funzione è di disegnare rettangoli. I valori indicati servono per fissare l'origine X,Y del disegno.

In questa annotazione, 00 indica l'angolo in alto a sinistra sullo schermo. Il valore X aumenta spostandosi orizzontalmente a destra: il valore Y aumenta spostandosi verso il basso.

La dimensione della griglia varia a seconda del *modo* grafico usato. In alta risoluzione, si hanno normalmente 320 pixel in larghezza e 200 in altezza. La risoluzione orizzontale si dimezza usando il modo multicolore, a 160 × 200 pixel. L'istruzione REC, che deve esser preceduta dal comando HIRES, necessità di più valori:

```
10 HIRES□ 10
20 REC 10,50,200,50,1
30 GOTO 10
```

In questo esempio, la prima coppia di numeri dopo la REC sono le coordinate X,Y (10 pixel da sinistra, 50 dal basso). Il successivo valore (200) è la lunghezza orizzontale del rettangolo, seguita dall'altezza (50). Tutti i valori sono in pixel.

Il valore finale, alla linea 20, definisce il 'tipo di plot' del Simon's BASIC. Qui, un 0 cancella, mentre un 1 disegna i punti. Il valore 2 *inverte* la condizione di un punto da *off* a *on* o viceversa.

L'istruzione MULTI è impiegata di supporto a HIRES, per attivare il funzionamento multicolore. Si possono selezionare tre colori, che vengono definiti da HIRES e selezionati agendo sul valore 'tipo' di plot' nelle successive istruzioni di disegno. Ecco un esempio:

```
10 HIRES□ 1,0
20 MULTI 7,8,14
30 REC 0,75,100,50,1
40 REC 70,50,50,50,2
50 REC 100,25,50,100,3
100 GOTO 10
```

Si noti come l'ultimo elemento di ogni REC (ossia il valore del 'tipo di plot') seleziona uno dei valori indicati da MULTI. È così che viene scelto uno dei colori disponibili, tra quelli predefiniti. Si provi a variare qualcuno dei valori finali alle linee 30, 40 e 50. Si ricordi che per selezionare il primo colore definito da MULTI si usa il valore 1, per il secondo 2, ecc. Un valore del 'tipo di plot' pari a 4, in modo MULTI, provoca l'inversione del tracciamento. I colori possono esser cambiati usando LOW COL, seguito da tre valori, ad esempio:

```
99 LOW COL 3,5,6
```

L'accesso a questi colori avviene in modo analogo a quello usato per REC, rispetto all'istruzione MULTI, nell'esempio precedente. Per tornare ai colori definiti in origine, l'istruzione da usare è HI COL, che non richiede l'uso di alcun valore. Mediante queste ultime due istruzioni, si possono alternare più volte, all'interno del medesimo programma, i colori impiegati nei disegni ad alta risoluzione.

I DISEGNI

Sono disponibili diverse istruzioni 'di disegno': PLOT, LINE, CIRCLE, ARC, ANGL e DRAW, tutte utilizzabili, anche combinate tra loro, per produrre effetti grafici sullo schermo. Tutte le istruzioni, eccetto DRAW, sono seguite da alcuni valori, per indicare le coordinate X, Y, necessarie. Vediamo per prime PLOT, LINE e CIRCLE.

PLOT serve per illuminare singoli punti (usare un NEW sul computer per rimuovere eventuali programmi precedenti):

```
10 HIRES□ 1,0
20 FOR X=0 TO 100 : FOR Y=0 TO 100
30 PLOT H,Y,1
40 FOR D=1 TO 10: NEXT D
50 NEXT Y,X
60 GOTO 10
```

Il programma traccia una linea partendo dall'angolo superiore sinistro. X e Y sono variate all'interno del ciclo FOR ... NEXT, che inizia alla linea 20. L'ultimo valore, nella 30, è quello del 'tipo di plot'.

Il Simon's BASIC offre un'utile istruzione, per introdurre pause nell'esecuzione: PAUSE. Possiamo sostituire la linea 40 con:

```
40 PAUSE 1
```

che crea una sosta di un secondo prima di passare a illuminare il punto successivo. PAUSE può essere usato anche per ritardare la conclusione del programma:

```
60 PAUSE 15
```

trattiene il programma in funzione per 15 secondi, prima di far apparire il simbolo di 'accettazione comandi' sullo schermo.

Forse, l'istruzione più utile al disegno è LINE, che serve per tracciare una linea tra due punti: basta indicare le coordinate X,Y dei due punti e il gioco è fatto! Si impartisca un NEW, e si trascriva il seguente programma:

```
10 COLOUR 3,0: HIRES 1,0: MULTI 8,3,6
40 LINE 55, 100,40,80,3
160 PAUSE 20
```

Si ottiene il tracciamento di una linea, nel terzo colore fissato da MULTI, che potrebbe formare parte di un grafico, aggiungendo altre linee. Il comando CIRCLE è seguito dalle coordinate X,Y del centro, dalla misura del raggio orizzontale e del raggio verticale e, dal valore del 'tipo di plot':

```
20 CIRCLE 65,100,10,16,2
```

Il cerchio è tracciato nel secondo colore specificato da MULTI. Si noti che le misure dei raggi differiscono: ciò serve a compensare il fatto che lo schermo è rettangolare, mentre i 'passi' di tracciamento, sono costanti. In Modo HIRES, il raggio Y (penultimo valore nell'istruzione CIRCLE) deve essere 1,4 volte il raggio X (il terzo valore indicato nella CIRCLE. Nel Modo MULTI, invece, il rapporto tra Y e X deve essere di 1,6, per ottenere un'adeguata curvatura del cerchio.

USO DI PAINT

Il comando PAINT è usato per colorare una sagoma, dal contorno chiuso, in una delle tinte definite dall'istruzione MULTI. Ecco un esempio:

```
30 PAINT 65,110,3
```

Questa linea colora il cerchio in azzurro, colore specificato dalla linea 10, immessa precedentemente. L'area da colorare deve essere completamente racchiusa in un perimetro, e le coordinate X,Y devono riferirsi a un punto interno a tale area. Si facciano degli esperimenti, variando i valori riportati nella linea 40: si può cambiare colorazione agendo sul valore del 'tipo di plot', ma si notino le conseguenze di qualche errore, introdotto deliberatamente, sui valori X ed Y delle coordinate.

Per arricchire il programma precedente, ottenendo un disegno, si aggiungano le seguenti linee:

```
50 LINE 40,80,0,120,3
60 LINE 0,120,40,100,3
70 LINE 40,100,55,105,3: PAINT50,100,2
80 LINE 70,100,110,70,3
90 LINE 110,70,150,90,3
100 LINE 150,90,110,90,3
110 LINE 110,90,75,100,3: PAINT90,95,2
120 LINE 60,110,50,130,3
130 LINE 70,105,80,130,3
140 LINE 50,130,80,130,3: PAINT60,120,2
150 FOR Z=63TO67:LINEZ,110,65,140,1:
    NEXT:PAUSE 10
```

L'immagine che si ottiene sullo schermo è quella di un uccello in volo. Si presti attenzione, in particolare, a come il valore del 'tipo di plot' venga più volte modificato durante l'esecuzione del programma.

In primo luogo, nel programma viene stabilito il colore dello schermo, in alta risoluzione, e quello del disegno (linea 10). Successivamente, la linea 20 disegna il cerchio che rappresenta la testa dell'uccello, che viene poi colorata dalla linea 30. Le linee da 40 a 70 provvedono a disegnare e colorare l'ala sinistra, le linee da 80 a 110 l'ala destra. La coda viene creata e colorata dalle linee da 120 a 140, mentre il becco viene prodotto con un interessante impiego del ciclo FOR ... NEXT nella linea 150. Infine, con una PAUSE, ci concediamo il tempo di ammirare il tutto.

Come nel Commodore 64, anche nel Vic 20, la versione standard del BASIC è priva di istruzioni che diano immediato accesso alle capacità grafiche del computer.

Una volta ancora, la miglior cosa è procurarsi la cartuccia d'espansione Vic Super Expander. Questa viene inserita nell'apposita fessura presente sul Vic 20, permettendo di ampliare la gamma dei comandi BASIC disponibili a quelli per la grafica. Accendendo l'apparecchio con la cartuccia inserita, si trascriva:

```
10 GRAPHIC 2: SCNCLR
20 FOR X=0 TO 100 STEP6:FOR Y=0 TO
   100 STEP6
30 POINT 1,X,Y
40 FOR D=1 TO 10:NEXT D
50 NEXT Y,X
60 GOTO 10
```

L'effetto di questo programma equivale a quello ottenuto con l'esempio riportato per il Commodore 64, nella pagina precedente. Anche sul Vic 20 si può ottenere il disegno di un uccello, simile a quello del Commodore 64. Ecco il programma:

```
10 GRAPHIC 2:COLOR 0,0,3,6
20 CIRCLE1,460,400,40,64
25 CIRCLE1,460,300,30,34
30 PAINT1,460,440
40 DRAW1,420,400TO360,320
50 DRAW1,360,320TO200,480
60 DRAW1,200,480TO360,400
70 DRAW1,360,400TO420,420:PAINT1,400,
   400
80 DRAW1,480,400TO640,280
90 DRAW1,640,280TO800,360
100 DRAW1,800,360TO640,360
110 DRAW1,640,360TO500,400:PAINT1,560,
    350:COLOR0,0,5,5
120 DRAW1,440,440TO400,520
130 DRAW1,480,420TO560,520
140 DRAW1,400,520TO560,520:PAINT1,440,
    480
150 DRAW1,480,300TO520,300
160 POKE 36865,39+RND(1)*2:GOTO 160
```

Una volta ancora, per le spiegazioni ci si può riferire a quanto detto per il programma del Commodore 64, benché il nome delle istruzioni sia diverso.

All'accensione, gli Acorn sono automaticamente predisposti in Modo text, perciò, prima di poter disegnare qualcosa, occorre selezionare uno dei modi 5 grafici a disposizione, indicati coi valori: 0, 1, 2, 4 e 5. Ciascuno di questi dispone di un differente numero di colori e di una diversa risoluzione. Per il momento usiamo il Modo 1: si digita sulla tastiera MODE 1 (terminando con RETURN).

Tutti i disegni vengono creati usando il cursore grafico, che può essere facilmente spostato sullo schermo grazie al comando MOVE X,Y. Il comando DRAW X,Y serve per tracciare linee, mentre con PLOT si ottiene una varietà di effetti.

Digitiamo ora DRAW 640, 512 e premiamo RETURN. Dovremmo vedere una linea che va dal margine inferiore dello schermo fino al centro. Ora si scriva DRAW 1279,0: ecco un'altra linea, che parte da dove termina la precedente e prosegue verso l'angolo in basso a destra. Un terzo comando completa il triangolo: DRAW 0,0.

Questo è stato piuttosto facile, ma tutti quei numeri generano un po' di confusione. In realtà non sono altro che le coordinate X,Y del punto che vogliamo 'raggiungere'. Lo schermo è suddiviso in tanti piccoli quadratini. In senso orizzontale, lungo l'asse X, ci sono 1280 quadratini (numerati da 0 a 1279), in senso verticale, lungo l'asse Y, ve ne sono 1024 (da 0 a 1023). Ecco perché il centro dello schermo ha per coordinate 640, 512.

FACCIAMO UN DISEGNO

Quando si intende riprodurre un disegno sullo schermo, conviene partire da un progetto realizzato su carta quadrettata, in modo da meglio individuare le coordinate dei vari punti. A questo scopo, il manuale dell'apparecchio, in appendice, fornisce un facsimile della quadrettatura da usare.

Volendo iniziare da un punto diverso da quello dell'origine, possiamo usare il comando MOVE X,Y, che serve per *muovere* il cursore, senza lasciar traccie visibili, in un particolare punto dello schermo con coordinate X e Y. Usando subito dopo un comando DRAW, la linea inizia propria da questo nuovo punto. Il miglior insegnamento è la pratica:

```
10 MODE 1
20 MOVE 300,200
30 DRAW 900,200
```

Eseguendo il programma, con un RUN, comparirà una linea vicina al margine inferiore dello schermo: è la base per il disegno di una casetta, che viene completata aggiungendo le seguenti linee:

```
40 DRAW 900,600
50 DRAW 300,600
60 DRAW 300,200
70 MOVE 300,600
80 DRAW 600,900
90 DRAW 900,600
```

Si provi, durante l'immissione, a dare un RUN prima di passare alla linea successiva: si può così apprezzare la costruzione *mattone su mattone* della casetta.

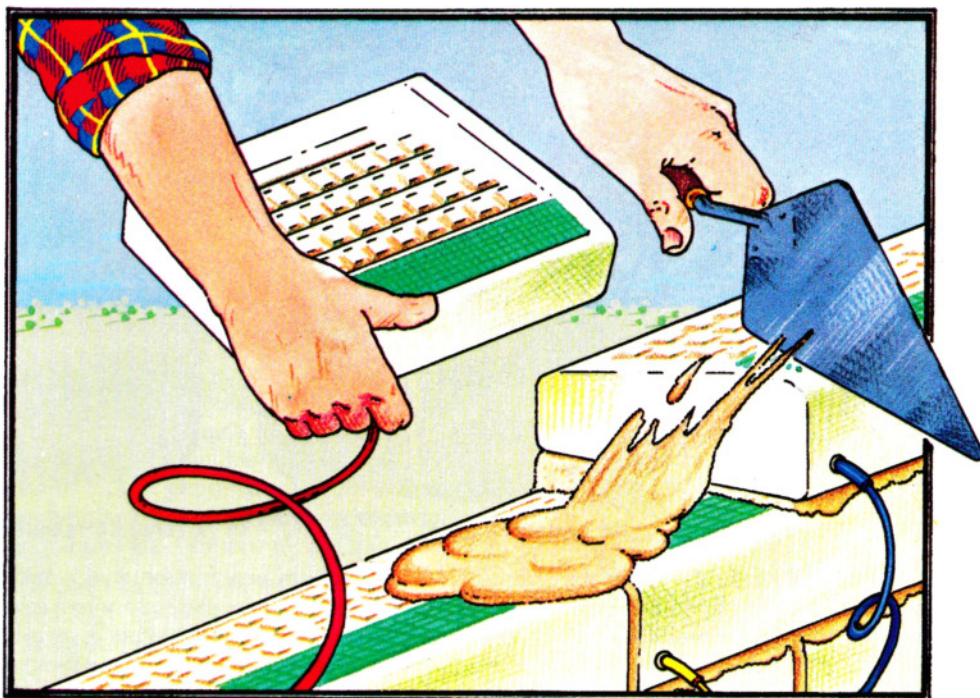
USO DEGLI ALTRI MODI

Fin qui, abbiamo operato esclusivamente usando il Modo 1. Si provi a cambiare, nella linea 10 del programma precedente, l'istruzione MODE 1 in MODE 5. Lanciando nuovamente il programma, si noterà che adesso i pixel sono di dimensioni molto maggiori rispetto ai precedenti, e che le linee sono molto più spesse, in particolare quelle che formano il tetto della casetta. Detto brevemente, il Modo 5 ha una risoluzione più bassa del Modo 1. Si provi, ancora, a cambiare la linea 10, specificando ora MODE 0: stiamo ora usando il modo grafico con la massima risoluzione ottenibile e il disegno è molto più preciso. Il Modo 2 ha la stessa risoluzione del Modo 5, il Modo 1 la stessa del Modo 4.

DISEGNI A COLORI

Se non viene specificato un particolare colore, i disegni risultano in bianco su sfondo nero. Per usare altre combinazioni, si ricorre al comando CGOL 0. Si trascuri, per adesso, il valore 0 qui indicato.

Adesso riportiamo la linea 10 alla formulazione originale, con MODE 1. Questo modo grafico dispone di quattro colori: il nero, il rosso, il giallo ed il bianco. Questi corrispondono, rispettivamente, ai valori 0, 1, 2, e 3. Se impartiamo il comando GCOL 0,2 e tracciamo una linea, questa sarà di colore giallo. Usando GCOL 0,1 la linea risulterà rossa. Si provi a modificare il programma che disegna la casetta, in modo da ottenere pareti gialle e il tetto rosso. La prima istruzione CGOL va collocata alla linea 15, e la seconda alla linea 75.

Anche la colorazione dello sfondo può esser modificata. Il comando è lo stesso, GCOL 0, e basta aggiungere 128 al valore che indica il colore. Per esempio, GCOL 0,129, seguito da CLG (che serve a ripulire

lo schermo grafico) produce uno sfondo rosso.

Il seguente programma, dimostrativo riempie lo schermo con rettangoli di dimensioni scelte a caso, variando il colore dello sfondo (giallo, rosso, bianco e nero):

```
10 MODE 1
15 FOR scatole= 1 TO 30
20 X=RND(1000) : Y=RND(800)
30 base=RND(4)*100 : lato=RND(4)*100
40 GCOL 0,RND(3)
50 MOVE X,Y
60 DRAW X+base,Y
70 DRAW X+base,Y+lato
80 DRAW X,Y+lato
90 DRAW X,Y
100 NEXT
```

La linea 15 inizia un ciclo che disegna 30 rettangoli. La linea 20 fissa l'angolo di partenza di ciascun rettangolo, la 30 sceglie le misure della base e dell'altezza, la 40 seleziona i colori, mentre le linee che seguono disegnano ciascun rettangolo.

Preferiamo una diversa combinazione di colori per il disegno, diciamo rettangoli bianchi e neri su sfondo rosso? Basta apportare un paio di modifiche al programma; con una, selezioniamo il rosso per lo sfondo:

```
13 GCOL 0,129: CLG
```

con l'altra, cambiamo la linea 40 in:

```
40 GCOL 0,(RND(2) - 1)*3
```

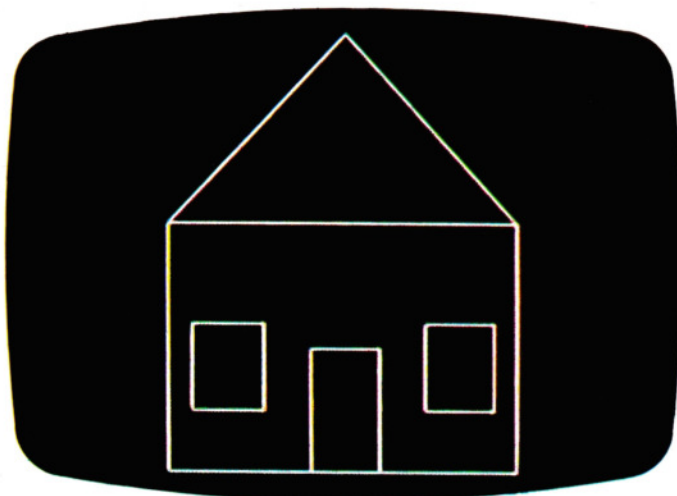
il che assicura che i colori corrisponderanno al nero (0) e al bianco (3).

USO DEL COMANDO PLOT

Abbiamo visto come muovere il cursore grafico sullo schermo, come tracciare linee e selezionare i colori. Ma l'uso di PLOT non è limitato a questo. Per esempio, il comando PLOT 69, X,Y serve ad illuminare un singolo punto sullo schermo mentre possiamo tracciare linee tratteggiate con PLOT 21, X,Y, come mostrato nel seguente programma:

```
10 MODE 1
20 GCOL 0,130 : CLG
30 PRINT TAB(0,3)"Per favore firmare"' "sulla
   riga punteggiata"
35 GCOL 0,0
40 MOVE 500,190
50 PLOT 21,1100,190
60 INPUT TAB(17,25),firma
```

Questo breve programma illustra anche come si possa mescolare grafica e testo sullo stesso schermo. Il testo viene posizionato usando l'istruzione TAB. Una variante molto utile del comando PLOT è PLOT 85, che disegna e colora un triangolo. Il formanto completo è: PLOT 85, X,Y e l'effetto che si ottiene è di tracciare un triangolo che ha per vertici il punto di coordinate X,Y e gli ultimi due punti *visitati* precedentemente al comando PLOT. Si provi quanto segue:

```
10 MODE 1
20 GCOL 0,2
30 MOVE 200,200
40 MOVE 1080,200
50 PLOT 85,640,900
```

3. Si può iniziare da queste semplici linee

4. Questo missile, sul BBC, è composto da 15 triangoli

Si potrà apprezzare che non viene usato il comando DRAW: ci si limita a *visitare (ossia posizionarsi) su due punti,* (linee 3Ø e 4Ø), per poi sfruttare le prestazioni del comando PLOT 85, che disegna e colora per noi il triangolo.

Con un poco di pratica, anche forme di una certa complessità si possono scomporre in più triangoli, facilitandone la rappresentazione sullo schermo del computer. Ad esempio, un rettangolo si può costruire con due triangoli, come avviene nel programma che segue:

```
10 MODE 1
20 GCOL 0,2
30 MOVE 300,300
40 MOVE 300,700
50 PLOT 85,1000,300
60 PLOT 85,1000,700
```

Infine, ecco un programma che fa buon uso del comando per creare triangoli colorati: viene prodotta l'immagine di un missile a tre stadi, al centro dello schermo, usando tre colori e 15 triangoli (figura 4):

```
10 MODE 1
20 MOVE 600,1000
30 MOVE 520,900
40 PLOT 85,680,900
50 GCOL 0,2
60 PLOT 85,520,800
70 PLOT 85,680,800
80 GCOL 0,3
90 PLOT 85,480,740
100 PLOT 85,720,740
110 GCOL 0,2
120 PLOT 85,480,540
130 PLOT 85,720,540
140 GCOL 0,3
150 PLOT 85, 40,460
160 PLOT 85,760,460
170 GCOL 0,2
180 PLOT 85,440,120
190 PLOT 85,760,120
200 GCOL 0,1
210 MOVE 720,120
220 MOVE 680,120
230 PLOT 85,760,20
240 PLOT 85,640,20
250 MOVE 520,120
260 MOVE 480,120
270 PLOT 85,560,20
280 PLOT 85,440,20
```

Ora ci si eserciti a creare immagini diverse, secondo la propria fantasia. Volendo una gamma più ampia di colori, si può passare al Modo 2, che ne offre ben 16.

All'accensione, il funzionamento dello schermo è selezionato automaticamente in Modo text. In questa condizione, si possono far apparire sullo schermo normali scritte o simboli grafici speciali, conservati in ROM e utilizzati da altri programmi.

Ma se desideriamo produrre disegni più elaborati, è necessario ricorrere al funzionamento in alta risoluzione, di cui l'apparecchio è dotato. Le immagini che si ottengono sono di effetto strabiliante, anche se non è possibile mescolare grafica e testo sullo stesso schermo.

TRACCIARE UNA LINEA

Ecco un programmino per disegnare due linee, col Modo grafico 3:

```
20 PMODE 3,1
30 PCLS
40 SCREEN 1,0
50 LINE (0,191)—(255,0),PSET
60 LINE (0,0)—(255,191),PSET
70 GOTO 70
```

Per prima cosa, prima di tracciare le linee, viene selezionato (alla linea 2Ø), il modo grafico e la pagina (nella memoria del computer) ove depositare le informazioni grafiche. La linea 2Ø contiene il comando PMODE 3,1, che comunica al computer di usare il Modo grafico 3, e di memorizzare il disegno che apparirà sullo schermo partendo dalla pagina grafica 1.

Successivamente, la linea 3Ø ripulisce lo schermo, approntandolo per un nuovo disegno in alta risoluzione. Il comando PCLS equivale, per la grafica, al normale CLS usato nel Modo text.

Il comando che attiva il funzionamento in alta risoluzione è SCREEN. Cosicché, la linea 4Ø contiene SCREEN 1,Ø. L'insieme di colori selezionato dal valore Ø, nel comando, comprende i colori: verde, giallo, blu e rosso. Specificando un valore 1, si otterrebbero i colori nocciola, azzurro, magenta e arancione.

Questo programma traccia delle linee in verde e in rosso (noti come colori *default*, ossia selezionati in mancanza di esplicite indicazioni). Per usare altre combinazioni di colori, si ricorre al comando COLOR, che sarà trattato più avanti.

La prima riga viene tracciata dalla linea 5Ø: i numeri tra parentesi indicano al computer la posizione dei due estremi della retta. Lo schermo in alta risoluzione è suddiviso in 256 × 192 pixel, numerati da Ø a 255 in senso orizzontale, e da Ø a 191 in senso verticale (dall'alto in basso).

Infine, PSET comunica alla macchina di usare il colore di valore massimo tra quelli dell'insieme selezionato. Nel nostro caso il colore è il rosso.

La linea 60 traccia la seconda linea e funziona come la 50, salvo che la posizione degli estremi è stata variata.

Per evitare che, una volta eseguito il programma, il computer ritorni al funzionamento in Modo text (quello per lui *normale* o *default*, cancellando completamente il disegno appena realizzato), è necessario utilizzare un ciclo senza fine, che blocchi l'esecuzione. Ciò si ottiene alla linea 70, che contiene una GOTO 70.

DISEGNARE CERCHI

Questa operazione è veramente semplice, sia sul Dragon che sul Tandy, perché esiste nel BASIC un apposito comando CIRCLE.

Il seguente programma disegna tre cerchi, di diverse dimensioni e colori:

```
20 PMODE 3,1
30 PCLS
40 SCREEN 1,0
50 CIRCLE (70,95),10,2
60 CIRCLE (118,95),20,3
70 CIRCLE (184,95),30,4
80 GOTO 80
```

Sia il modo grafico, che l'insieme di colori sono quelli già usati in precedenza.

Il comando CIRCLE, alle linee 50, 60 e 70 contiene quattro elementi: il primo e il secondo sono la posizione del centro, il terzo è la lunghezza del raggio in pixel e il quarto corrisponde al colore da usare.

Il cerchio disegnato dalla linea 50 ha centro in (70, 95), un raggio di 10 pixel ed è colorato in giallo. Il cerchio della linea 60 è blu, ha un raggio di 20, con centro in (118,95). Il terzo cerchio è rosso, ha un raggio di 30 e centro in (184,95).

Infine, la linea 80 crea un ciclo per evitare che il computer, tornando al comando di funzionamento normale, cancelli il disegno ottenuto in alta risoluzione.

CREAZIONE DI UNA JEEP

Si trascriva e si esegua il seguente programma. Il risultato assomiglia a quanto riportato in figura 5, ma senza molti dei particolari che vi figurano.

```
20 PMODE 3,1
30 PCLS
40 SCREEN 1,0
50 LINE (108,64)—(188,64),PSET
60 LINE —(188,116),PSET
70 LINE —(104,116),PSET
80 LINE —(96,96),PSET
90 LINE —(70,96),PSET
110 LINE —(74,96),PSET
120 LINE —(74,86),PSET
130 LINE —(114,86),PSET
140 LINE —(132,104),PSET
150 LINE —(140,104),PSET
160 LINE —(140,64),PSET
170 LINE(76,96)—(76,108),PSET
180 LINE —(100,108),PSET
320 GOTO 320
```

Come in precedenza, il modo selezionato è il 3. La linea 50 inizia il disegno dalla posizione (108, 64), tracciando una linea rossa fino al punto (188, 64). Per continuare il disegno dallo stesso punto, basta usare una nuova istruzione LINE.

Le linee da 60 a 160, dunque, non fanno altro che definire i punti 'd'arrivo' di ciascuna nuova linea.

La linea 170, al contrario, deve tracciare una linea partendo da un punto totalmente nuovo, quindi occorre definire sia questo, che il punto 'di arrivo'. Adesso, aggiungiamo i dettagli (migliori contorni, un parabrezza, i copertoni, ecc.):

```
200 LINE(140,88)—(188,88),PSET
220 LINE(118,64)—(104,86),PSET
230 CIRCLE(82,116),14,3
250 CIRCLE(82,116),6,2
270 CIRCLE(168,116),14,3
290 CIRCLE(168,116),6,2
```

La linea 200 completa la carrozzeria della jeep, la linea 220 disegna il parabrezza. Nelle linee 230 e 270, CIRCLE viene usata per tracciare i copertoni, e nelle linee 250 e 290 per tracciare la sagoma dei cerchioni delle ruote.

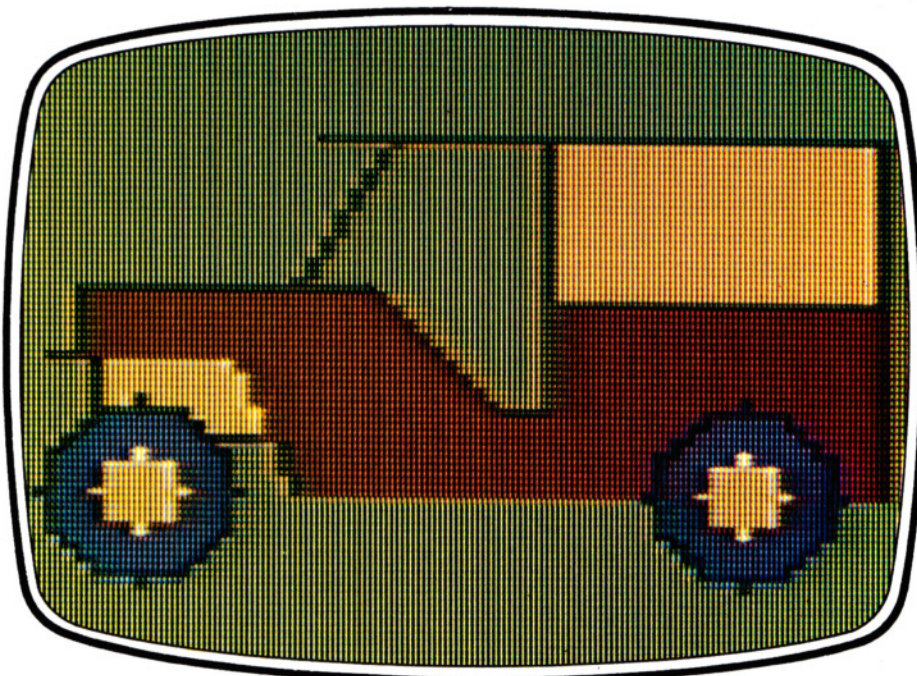
COLORIAMO I DISEGNI

È facile *verniciare* la jeep: basta usare il comando PAINT e ci pensa il computer.

Il comando PAINT riempie una figura, il cui perimetro sia chiuso, con il colore selezionato a tal scopo. I valori da specificare sono quattro; è facile intuire che i primi due sono la posizione del punto dal quale iniziare la colorazione, il terzo è il codice del colore e il quarto è il colore del bordo o contorno, dove l'operazione deve fermarsi. Si aggiungano le seguenti linee al programma precedente, e si vedrà come funziona, in pratica, il comando PAINT:

```
190 PAINT(90,100),2,4
210 PAINT(120,110),4,4
240 PAINT(82,116),3,3
260 PAINT(82,116),2,2
280 PAINT(168,116)3,3
300 PAINT(168,116),2,2
310 PAINT(180,80),2,4
```

Si esegua il programma con un RUN, e vedremo le varie parti della jeep colorarsi come abbiamo chiesto.

Non è possibile raggruppare le linee di colorazione in fondo al programma per la semplice ragione che l'area da colorare deve essere contornata da uno stesso colore o dal margine dello schermo, se no il computer si confonde e sconfina nella colorazione. I colori disponibili con PMODE 3,1 sono: verde (1), giallo (2), blu (3) e rosso (4). Volendo colorare la jeep diversamente, basta definire un altro gruppo di colori nelle varie linee contenenti l'istruzione PAINT. Inoltre, si può arricchire la jeep con altri particolari (ad esempio, un volante), mediante l'impiego di nuove LINE e CIRCLE. Buon divertimento!

5. Una jeep ottenuta con i semplici comandi del Dragon

SVELIAMO I MISTERI DELLE VARIABILI

Tutte quelle X e Y nei programmi possono confondere. Spieghiamo che cosa sono e a cosa servono le variabili numeriche e alfanumeriche. Il loro impiego nei programmi

Quando si desidera memorizzare un'informazione nel computer, è necessario comunicargli come identificarla, altrimenti la macchina non saprà cosa fare con l'informazione, né saprà ritrovarla in seguito.

Il tipo più frequente d'informazione è quello numerico: vengono immessi uno o più numeri, il cui valore può variare durante l'esecuzione del programma. Uno di questi numeri potrebbe rappresentare, ad esempio, il punteggio in un gioco oppure il numero di posizioni che ha percorso un carattere sullo schermo.

Spesso, comunque, si tratta semplicemente di un numero il cui valore cambia durante la ripetizione di una sezione di programma. Ecco un esempio molto comune:

```
10 LET X=Ø
20 LET X=X+1
30 PRINT X; "□";
40 FOR T=Ø TO 1Ø: NEXT T
50 GOTO 20
```

In questo caso, la linea 5Ø crea un ciclo che fa ripetere le linee da 2Ø a 5Ø un numero infinito di volte: ad ogni passaggio dalla linea 2Ø, il numero rappresentato con X (al quale, originariamente, era stato assegnato il valore Ø nella linea 1Ø) viene aumentato di un'unità.

La X di questo programma viene chiamata *variabile numerica*, poiché contiene un numero.

Un modo per comprendere il funzionamento di una variabile numerica è quello di immaginarsi una serie di scatole o degli scomparti, simili a quelli usati per le chiavi negli alberghi. Ciascuna scatola o casella viene identificata con un nome. Nei BASIC più semplici, questo nome è costituito da una semplice lettera alfabetica. Quando vogliamo memorizzare un valore, lo assegniamo a uno dei contenitori. Per esempio:

```
LET C=25
```

Questa operazione deposita il valore 25 nella variabile C. Non tutti i BASIC richiedono l'uso della parola LET (e viene spesso omessa, infatti). Finché si è alle prime armi nella programmazione, comunque, conviene usare LET per non confondersi quando si rileggono i listati dei programmi.

Una volta depositato, il valore può essere utilizzato in altre linee di programma per eseguire un calcolo, ad esempio.

Possiamo adesso scrivere:

```
PRINT C*4
```

oppure:

```
PRINT C/5
```

Questi comandi danno come risultato, rispettivamente, 1ØØ e 5. Ma se adesso digitiamo:

```
PRINT C
```

vedremo che il valore di C è inalterato. Per farlo aumentare a 3Ø, per esempio, possiamo usare più metodi:

```
LET C=30
```

oppure:

```
LET C=C+5
```

Quest'ultima forma corrisponde esattamente a quanto usato nel primo programmino illustrato.

I NOMI DELLE VARIABILI

Le combinazioni di lettere e simboli consentite nei nomi delle variabili numeriche variano da una versione all'altra di BASIC. Le principali indicazioni in merito sono riportate nella tabella a pagina 95.

Una cosa certa è che il nome di una variabile non può mai iniziare con un numero e sarebbe assurdo scrivere:

```
LET 7=14
```

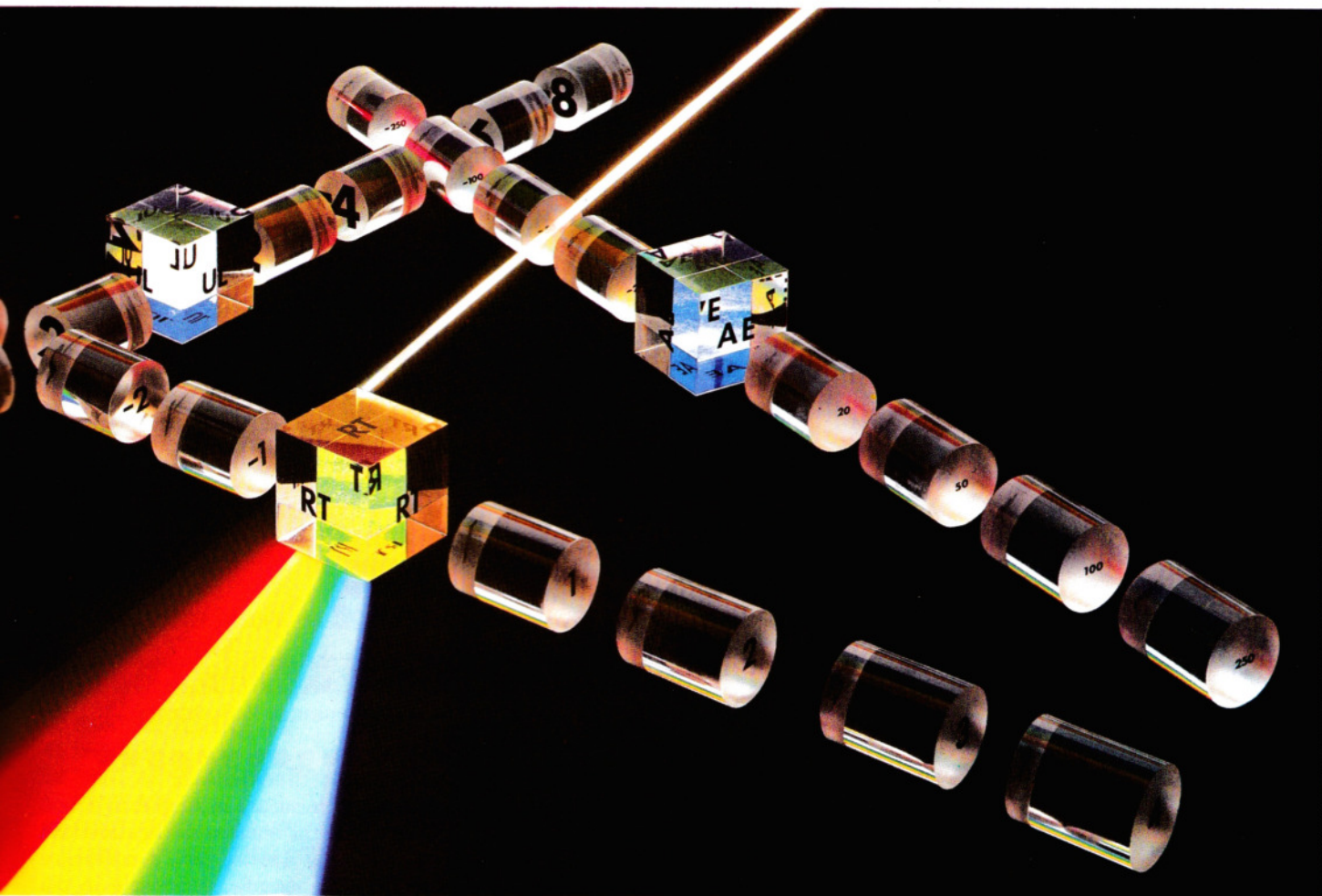
Un'altra restrizione è che non si possono usare gli stessi nomi adottati per i comandi e le istruzioni BASIC, ad esempio: AND, THEN o GOTO.

VARIABILI IN AZIONE

Tutti i programmi contengono più variabili, spesso intercollegate da un calcolo o un'operazione logica. Ecco, a titolo d'esempio, un breve programma che simula il funzionamento di un distributore di benzina.

```
10 LET litri =Ø
20 LET lire=Ø
40 PRINT "Benzina normale (1) o Super (2)?"
50 INPUT a
60 if a=1 THEN LET qta=1/22.22
70 if a=2 THEN LET qta=1/25
80 if a<1 OR a>2 THEN GOTO 50
90 CLS
100 IF INKEY$="n" THEN LET lire=lire
    +5Ø:LET litri=litri+qta
110 PRINT AT 1Ø,12;"LITRI:";litri;"□□□□
    □"
130 PRINT AT 12,12;"LIRE:□";lire;"□□□"
150 IF INKEY$="f" THEN CLS:GOTO 1Ø
200 GOTO 1ØØ
```

```
10 LET LITRI=Ø
```

- COS'È UNA VARIABILE?
- COME SI USANO LE VARIABILI
- LE VARIABILI DI CONTROLLO NEI CICLI **FOR ... NEXT**
- COS'È UNA STRINGA?
- USO DELLE VARIABILI NUMERICHE E DELLE STRINGHE
- COS'È UNA STRINGA NULLA?
- USO DELLE STRINGHE NULLE NEI PROGRAMMI DI GIOCO

```
20 LET LIRE=0
40 PRINT "BENZINA NORMALE (1) O SUPER
   (2)?"
50 INPUT A
60 IF A=1 THEN LET QTA=1/22.22
70 IF A=2 THEN LET QTA=1/25
80 IF A<1 OR A>2 THEN GOTO 40
90 CLS
100 IF INKEY$<>"N" THEN GOTO 110
102 LET LIRE=LIRE+50
103 LET LITRI=LITRI+QTA
110 PRINT AT 9,12;"LITRI:";LITRI;"□□□□
    □"
120 PRINT AT 13,10;'LIRE:□";LIRE;"□□□□
    □"
130 IF INKEY$="F" THEN RUN
200 GOTO 100
```

```
10 LET LITRI=0
20 LET LIRE=0
30 PRINT "♡▣":POKE650,128
40 PRINT "BENZINA NORMALE (1) O SUPER
   (2)?"
50 INPUT A
60 IF A=1 THEN LET QTA=1/22.22
70 IF A=2 THEN LET QTA=1/25
80 IF A<1 OR A>2 THEN GOTO 40
90 PRINT "♡"
110 PRINT "▤▣▣▣▣▣▣▣
    ▐▐▐▐▐▐ LITRI"
120 PRINT "▤▣▣▣▣▣▣▣▣▣
    ▣▣▐▐▐▐▐▐ LIRE"
130 GET K$
140 IF K$="F" THEN PRINT "♡": GOTO 10
150 IF K$="N" THEN LET LIRE=LIRE+
    50:LET LITRI=LITRI+QTA
160 PRINT "▤▣▣▣▣▣▣▣▣▣
    ▐▐▐▐▐▐";LITRI
170 PRINT "▤▣▣▣▣▣▣▣▣▣
    ▣▣▣▣▐▐▐▐▐▐";LIRE
200 GOTO 130
```

```
10 LET litri=0
20 LET lire=0
25 @%=&90A
30 CLS: PRINT
40 PRINT "Benzina Normale (1) o Super (2)?"
50 INPUT a
60 if a=1 THEN LET qta=1/22.22
```

```
7Ø if a=2 THEN LET qta=1/25
8Ø if a<1 OR a>2 THEN GOTO 5Ø
9Ø CLS
1ØØ PRINT TAB(16,9);"LITRI:"
11Ø PRINT TAB(16,12);"LIRE:"
125 @%=&2Ø2ØA
13Ø K$=GET$
14Ø IF K$="F" THEN CLS:GOTO 1Ø
15Ø IF K$="N" THEN LET lire=lire+5Ø:
   LET litri=litri+qta
16Ø PRINT TAB(16,1Ø); litri
17Ø PRINT TAB(16, 13);lire
2ØØ GOTO 1ØØ
```

```
1Ø LET LITRI=Ø
2Ø LET LIRE=Ø
3Ø CLS:PRINT
4Ø PRINT "BENZINA NORMALE (1) O SUPER
   (2)?"
5Ø INPUT A
6Ø IF A=1 THEN LET QTA=1/22.22
7Ø IF A=2 THEN LET QTA=1/25
8Ø IF A<1 OR A>2 THEN GOTO 4Ø
9Ø CLS
1ØØ PRINT@ 2ØØ,"LITRI:"
11Ø PRINT@ 267,"LIRE :"
13Ø LET K=PEEK(344)
14Ø IF K=251 THEN CLS:GOTO 1Ø
15Ø IF K=247 THEN LET LIRE=LIRE+5Ø:
   LET LITRI=LITRI+QTA
16Ø PRINT@ 2Ø6,LITRI
17Ø PRINT@ 33Ø,LIRE
2ØØ GOTO 13Ø
```

Vale la pena di esaminare attentamente il listato di ciascun programma, sia per comprendere il funzionamento delle variabili, sia per vedere come sono impostati gli algoritmi di calcolo nei quali esse vengono impiegate. L'utilità di alcune appare subito evidente. La variabile a, per esempio, contiene il numero da noi scelto, corrispondente al tipo di benzina richiesto. Mentre i valori assegnati alla variabile QTA possono sembrare a tutta prima astrusi: in realtà si tratta della quantità di benzina corrispondente a 5Ø lire.

Appena si preme il tasto [I] (che stà per Inizio), la pompa comincia a *erogare* benzina del tipo prescelto, comunicando 'a scatti', sia la quantità che il prezzo totali. All'inizio del programma, le variabili LIRE e LITRI vengono azzerate e successivamente, finché si tiene premuto il tasto [T], ad esse viene sommato, progressivamente, l'incremento in lire e in litri relativo all'erogazione. Ogni iterazione non riporta l'esecuzione alla linea 1Ø (che azzererebbe i totali), ma alla 13Ø (1ØØ sullo Spectrum). Mantenendo in funzione il programma per un periodo abbastanza lungo, può capitare che i valori relativi ai litri erogati contengano svariati decimali. Ciò

perché non abbiamo specificato quanti ne vanno visualizzati. Questo aspetto sarà ripreso in lezioni successive.

VARIABILI DI CONTROLLO

Le variabili numeriche vengono adoperate, oltreché nelle frasi LET, anche nei cicli FOR ... NEXT, (vedere alle pagine 16-21). In questo caso sono dette *variabili di controllo*. Su alcuni computer, i nomi consentiti per le variabili di controllo differiscono da quelli per le variabili numeriche (si veda la tabella). Sullo Spectrum, ad esempio, si può scrivere:

```
LET punti=Ø
LET massimo=Ø
```

ma non si può adottare lo stesso criterio di chiarezza per quanto riguarda le variabili di controllo. È sbagliato scrivere:

```
FOR Tempo=999999
```

perché è consentito usare una sola lettera.

LE STRINGHE

Le stringhe, in BASIC, funzionano come borse da spesa: vi si possono depositare svariate informazioni, per poi manipolarle ed elaborarle singolarmente o globalmente. Le stringhe possono contenere un po' di tutto: lettere, numeri, spazi o simboli grafici. L'unico accorgimento è di racchiudere sempre ogni sequenza di caratteri tra due doppi apici ossia un " all'inizio e un " alla fine della sequenza.

Ecco alcuni esempi:

```
"NOME E INDIRIZZO DEL CLIENTE"
"IMMETTERE LA RISPOSTA"
"24 gennaio"
"Ø2-455 755"
"CAP ØØ1ØØ"
```

La stringa che segue è composta da simboli grafici (la si può ottenere direttamente soltanto usando lo Spectrum):

"■■■■■■■■■■■■■■■"

Il BASIC può misurare la lunghezza di una stringa (ossia il numero di caratteri che essa contiene). Inoltre, può concatenare due stringhe assieme, sezionarle in più pezzi o anche estrarre un solo carattere da una stringa. In effetti, il *contenuto* della stringa non interessa al computer: non è possibile applicare operazioni aritmetiche al contenuto di due stringhe. A prova di ciò, si trascriva e si lanci questo programma:

```
1Ø PRINT "2+2=□";
2Ø FOR N=1 TO 2ØØ: NEXT N
3Ø PRINT 1+2*2
```

```
1Ø PRINT "2+2=□";
2Ø FOR N=1 TO 4Ø
25 NEXT N
3Ø PRINT 1+2*2
```

Come si vedrà, il contenuto delle stringhe viene visualizzato così come immesso. Solo alla linea 2Ø viene fatto un calcolo e ne viene presentato il risultato. La linea 2Ø, in effetti, serve solo per dar l'impressione che il computer *stia pensando* alla risposta. Sull'Acorn, cambiare il valore 2ØØ in 2ØØØ allunga un poco l'attesa.

Se si desidera usare una stringa una sola volta in un programma, allora quanto visto è sufficiente. Ma, desiderando riutilizzarla più volte, si risparmia tempo e fatica (oltre che memoria!) assegnando un nome alla stringa, ossia creando una *variabile stringa*. La lunghezza massima del nome varia da un computer all'altro (vedere la tabella a lato), ma il nome deve essere sempre e comunque seguito dal simbolo del dollaro: $. Ecco ad esempio, un programma che accetta ordinazioni:

```
10 LET A$="PER FAVORE, INDICA QUANT"
20 LET B$="VUOI"
30 PRINT A$;"I□ HAMBURGER";B$: INPUT H
40 PRINT A$;"E□ PIZZE";B$: INPUT P
50 PRINT A$;"E□ ARANCIATE";B$: INPUT A
60 PRINT "GRAZIE. PAGARE";(H*200+P*
   1500+A*1200);" "□ALLA CASSA."
```

```
10 LET A$="PER FAVORE, INDICA QUANT"
20 LET B$="□VOUI"
30 PRINT A$;"I□HAMBURGER ';B$;
35 INPUT H
40 PRINT A$;"E□PIZZE";B$:
45 INPUT P
50 PRINT A$;"E□ARANCIATE";B$:
55 INPUT A
60 PRINT "GRAZIE. PAGARE";(H*2000+P*
   1500+A*1200);"□ALLA CASSA."
```

Non si tratta di un programma molto sofisticato, tuttavia rende abbastanza bene l'idea di come l'uso delle stringhe possa facilitare la programmazione (salvo, forse, nello ZX81).

Ecco perché i programmi di *word processing* e di contabilità, nei quali scritte del tipo 'Conto del Cliente', 'Prezzo unitario' o 'imponibile I.V.A.' ricorrono con grande frequenza, usano molto le variabili stringa.

Queste sono molto utili anche nella programmazione dei giochi. Infatti, se dobbiamo definire un 'muro' o la forma di un drago, ad esempio, può essere molto più comodo farlo assegnando alla sequenza di caratteri un nome, e usare tale nome nel resto del programma.

Inoltre, una variabile stringa può essere spostata a piacimento sullo schermo, come dimostra il seguente programma:

Le variabili: cosa si può e cosa non si può usare

Tipo variabile			
Variabile numerica	Lunghezza: nessun limite Sullo Spectrum, sono consentite maiuscole o minuscole, ma il computer non fa alcuna distinzione tra i due modi Punteggiatura non consentita Spazi: consentiti ma ignorati	Lunghezza massima: 255 caratteri Maiuscole o minuscole consentite, ma, nel primo caso, non devono iniziare con parole chiave (ad es.: TO come in TOTALE) Punteggiatura: solo la sottolineatura è consentita Spazi: non consentiti	Lunghezza massima: 255 caratteri, ma soltanto i primi due sono distinti Solo le masciuole sono consentite e non devono iniziare con parole chiave (ad. es. TO come in TOTALE) Punteggiatura non consentita Spazi: consentiti ma ignorati
Variabile di controllo in cicli FOR ... NEXT	Solo singole lettere (ad. es.: A)	Come per le variabili numeriche	Lettera singola oppure lettera singola seguita da un carattere (ad es. A, A3)
Variabili stringa	Solo singole lettere seguite da $ (ad es.: A$)	Come nelle variabili numeriche, ma seguite da $ come in INDIRIZZO$	Come per le variabili numeriche, ma seguite da $ (ad. es.: AD$)

```
10 LET b$="□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□
   □"
20 LET a$=b$+"□□□□BUONA PASQUA
   A TUTTI□□□"+b$
30 FOR n=1 TO 65
40 PRINT INK 2; AT 8, Ø; a$
   (n TO n+31)
50 PRINT INK 2; AT 12, Ø; a$ (66-n TO
   97-n)
60 BEEP .02,n/2
70 NEXT n
80 GOTO 30
```

```
4 MODE4
```

```
5 VDU23;8202;0;0;0;
6 VDU19,0,4;0;
7 VDU19,1,3;0;
8 CLS
10 LET B$="□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□□
   □□□□□□□□□"
20 LET A$=B$+"BUONA PASQUA A
   TUTTI"+B$
30 FOR N=1 TO 65
40 PRINT TAB(0,12);MID$(A$,N,40)
50 PRINT TAB(0,16);MID$(A$,65-N,40)
60 SOUND1,-10,N*12,1
65 D=INKEY(10)
70 NEXT N
80 GOTO 30
```

```
10 LET B$="□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□□
   □"
20 LET A$=B$+"□□□□BUONA PASQUA
   A TUTTI□□□"+B$
30 FOR N N=1 TO 65
40 PRINT AT 8, 0; A$ (N TO N+31)
50 PRINT AT 12, 0; A$ (66-N TO 97-N)
70 NEXT N
80 GOTO 30
```

```
5 CLS
10 LET B$="□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□□
   □"
20 LET A$.B$+"□□□BUONA PASQUA
   A TUTTI□□□"+B$
30 FOR N=1 TO 65
40 PRINT@ 160,MID$(A$,N,32)
50 PRINT@ 320,MID$(A$,66-N,32)
60 SOUND N*3,1: NEXT N
80 GOTO 30
```

```
10 PRINT "♥"
15 POKE 54277,33:POKE 54278,255:
   POKE 54273+23,15:POKE 54276,33
2 LET B$="□□□□□BUONA PASQUA A
   TUTTI□□□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□□□
   □":B$=A$
40 LET A$=RIGHT$(A$,61)+LEFT$(A$,1)
45 PRINT "▒⊟◘◘◘◘◘◘◘◘◘"
LEFT$(A$,40)
50 LET B$=RIGHT$(B$,1)+LEFT$(B$,61)
55 PRINT "π ◘ ◘ ◘ ◘ ◘
   ◘"LEFT$ (B$,40)
60 LET S=S+5:IFS>255THENS=0
70 POKE 54273,S
80 GOTO 40
```

```
10 PRINT"♥"
15 POKE 36878,15:S=127
20 A$="□□□□□BUONA PASQUA A
   TUTTI□□□□□□□□□□□□□□□□
   □□□□□□□□□□□□□□□□□□□□
   □":B$=A$
40 A$=RIGHT$(A$,61)+LEFT$(A$,1)
45 PRINT"▒⊟◘◘◘◘◘◘◘◘◘"
   LEFT$(A$,22)
50 B$=RIGHT$(B$,1)+LEFT$(B$,61)
55 PRINT"π◘◘◘◘◘◘◘"LEFT$
   (B$,22)
60 LET S=S+5:IF S>255THENS=128
70 POKE 36876,S
80 GOTO40
```

In questo caso, come si rileva dalla linea 20, si ha una stringa di notevoli dimensioni: tutti gli spazi della linea 10 (20 nei programmi Commodore), più il messaggio e più, inoltre, un'altra sequenza di caratteri. il tutto viene concatenato in una sola enorme stringa. Come eseguire questa ed altre manipolazioni di stringhe, verrà spiegato più avanti nel corso. Se, comunque, volessimo nel frattempo modificare il messaggio, ci si assicuri che la sua lunghezza corrisponda esattamente, altrimenti occorrerà modificare i valori contenuti nelle linee 40 e 55.

Un uso ancora più comune delle variabili stringa è durante l'immissione di informazioni il cui contenuto varia di volta in volta. Ad esempio:

```
10 PRINT "COME TI CHIAMI"
20 INPUT N$
30 PRINT "CIAO,□"; N$
```

In questo programma, la linea 10 visualizza (con una PRINT) il messaggio COME TI CHIAMI, seguito da un punto interrogativo. Il computer attende, quindi, che l'operatore immetta un nome al quale viene assegnata l'etichetta N$. Alla linea 30 viene visualizzato un 'CIAO', seguito dal contenuto di N$, immesso nella linea precedente. Sullo schermo appare dunque:

CIAO, GUIDO (o MARCO o GINO...)

o ogni cosa immessa alla linea 20.

Ai messaggi COME TI CHIAMI? e CIAO non viene assegnato alcun nome (come A$, COSA$, ecc.) poiché, essendo impiegati un sola volta nel programma, non è necessario che il computer se li rammenti.

Al contrario, è necessario attribuire un nome al dato che si immette (in questo caso il proprio nome), poiché può esser usato un nome diverso a ogni nuova esecuzione del programma. Inoltre, deve essere memorizzato (alla linea 20) per poi essere richiamato e visualizzato alla 30.

Per inciso, una versione più compatta del programmino dell'ultimo esempio è:

```
10 INPUT "COME TI CHIAMI";N$
20 PRINT "CIAO,□";N$
```

(Sugli Acorn cambiare ; in , alla linea 10).

LE STRINGHE NULLE

Ancora un altro uso delle stringhe si è incontrato nella sezione Giochi al computer (pagine 54-59). Esistono più varianti che dipendono dal modello di computer, ma più o meno assomiglia a:

```
20 IF A$="" THEN GOTO 10
```

I due doppi apici, in questo caso, non racchiudono un bel niente, ossia una *stringa nulla*. Il significato della linea è abbastanza evidente: se l'input è pari a nulla (ovvero se nessun tasto viene premuto), allora torna alla linea 10 e aspetta finché non viene digitato un carattere.

Simili linee di programma servono affinché l'esecuzione si blocchi in attesa di un comando, invece di correre velocemente alla linea successiva.

Un caso diverso è quello nel quale tra i due doppi apici sia racchiuso uno spazio. Se si modifica la linea precedente in:

```
20 IF A$="□" THEN GOTO 10
```

l'esecuzione ritorna alla linea 10 se, e soltanto se, il carattere immesso equivale a uno spazio (ossia se l'operatore preme il tasto spazio o la barra spaziatrice).

INDICE CUMULATIVO DEI FASCICOLI

NEL PROSSIMO NUMERO

☐ *Un gioco non è completo se mancano un CONTAPUNTI e un CONTATEMPO, che rendono l'azione più competitiva e avvincente. Ecco come realizzarli.*

☐ *Il computer è capace di elaborare enormi quantità di dati. Per immetterli, possiamo usare le comode istruzioni READ e DATA.*

☐ *Se ci capita spesso di scrivere molte lettere, con il nostro 'TEXT EDITOR' i risultati saranno ordinati e di sicuro effetto.*

☐ *Comprendere come il computer possa contare o eseguire calcoli è molto importante. Un primo passo è imparare cos'è la NUMERAZIONE BINARIA e come usarla.*

☐ *Presentare o richiedere le informazioni sullo schermo con ordine e precisione è semplice con i comandi PRINT e INPUT.*